

MARCELO RAMON DE VILLA

UML 2.0

INFORMAÇÃO E COMUNICAÇÃO



A expansão do Ensino Técnico no Brasil, fator importante para melhoria de nossos recursos humanos, é um dos pilares do desenvolvimento do País. Esse objetivo, dos governos estaduais e federal, visa à melhoria da competitividade de nossos produtos e serviços, vis-à-vis com os dos países com os quais mantemos relações comerciais.

Em São Paulo, nos últimos anos, o governo estadual tem investido de forma contínua na ampliação e melhoria da sua rede de escolas técnicas - Etecs e Classes Descentralizadas (fruto de parcerias com a Secretaria Estadual de Educação e com Prefeituras). Esse esforço fez com que, de agosto de 2008 a 2011, as matrículas do Ensino Técnico (concomitante, subsequente e integrado, presencial e a distância) evoluíssem de 92.578 para 162.105. Em 2016, no primeiro semestre, somam 186.619.

A garantia da boa qualidade da educação profissional desses milhares de jovens e de trabalhadores requer investimentos em reformas, instalações, laboratórios, material didático e, principalmente, atualização técnica e pedagógica de professores e gestores escolares.

A parceria do Governo Federal com o Estado de São Paulo, firmada por intermédio do Programa Brasil Profissionalizado, é um apoio significativo para que a oferta pública de Ensino Técnico em São Paulo cresça com a qualidade atual e possa contribuir para o desenvolvimento econômico e social do Estado e, consequentemente, do País.

Almério Melquíades de Araújo
Coordenador do Ensino Médio e Técnico



Centro Estadual de Educação Tecnológica Paula Souza

Diretora Superintendente

Laura Laganá

Vice-Diretor Superintendente

César Silva

Chefe de Gabinete da Superintendência

Luiz Carlos Quadrelli

REALIZAÇÃO

Unidade do Ensino Médio e Técnico

Coordenador

Almério Melquíades de Araújo

Centro de Capacitação Técnica, Pedagógica e de Gestão - Cetec Capacitações

Responsável

Lucília dos Anjos Felgueiras Guerra

Responsável Brasil Profissionalizado

Silvana Maria Brenha Ribeiro

Professores Coordenadores de Projetos

Carlos Eduardo Ribeiro

Fabricio Braoios Azevedo

Tiago Jesus de Souza

Autor

Marcelo Ramon de Villa

UML 2.0

Administração Central

Cetec Capacitações

Sumário

Cap. 1 - Introdução	5
O que é UML ?	5
Evolução da UML.....	5
Entendendo a classificação conceitual da UML	6
Entendendo a aplicação da UML na elaboração de software	7
Intercambio de diagrama, infraestrutura e Superestrutura UML	8
Cap. 2 – Modelagem Unificada UML.....	9
Diagramas estruturais e comportamentais.....	9
Classificando os diagramas.....	12
As principais regras para modelagem	14
Cap. 3 Diagrama de Classe	15
Classes	16
Objetos	18
Atributos.....	19
Métodos	36
Classes Abstratas.....	36
Relações	37
Dependência	37
Associação de Classes.....	44
Qualificadores de Associação	44
Interfaces	45
Cap. 4 Diagramas de Sequência	51
Quadros de Interação.....	51
Linhas de Vida em Diagramas UML.....	52
Mensagens em Diagramas UML.....	55

Administração Central

Cetec Capacitações

Fragmentos Combinados nos Diagramas de Sequência.....	61
Usos da Interação nos Diagramas de Sequência.....	61
Cap. 5 Diagramas de Pacote.....	66
Representação	66
Visibilidade	68
Importando e Acessando Pacotes	69
Mixagem de Pacotes	70
Estruturas compostas.....	72
Conectores	73
Cap. 6 Diagramas de Componentes	79
Instancias de Componentes	80
Artefatos	81
Conectores de montagem	82
Compartilhamento de Componentes.....	84
Portas e conectores.....	87
Estereótipos de Componentes	89
Cap. 7 Diagramas de Casos de Uso.....	91
Caso de Uso	93
Agentes	94
Pacotes	95
Subsistemas.....	96
Limites do Sistema	96
Modelagem Avançada de Casos de Uso.....	99
Cap. 8 Máquina de Estado.....	101
Comportamento Dirigido por Eventos	103
Estado de Submáquina.....	104
Criando Transições entre Estados	106
Estados	106

Administração Central

Cetec Capacitações

Transições.....	107
Cap. 9 Diagrama de Atividade	109
Diagramas de atividade de leitura.....	109
Fluxos de dados.....	115

Administração Central

Cetec Capacitações

CURSO DE UML – UNIFIED MODELING LANGUAGE

Cap. 1 - Introdução

O que é UML ?

É um acrônimo para a expressão *Unified Modeling Language*, basicamente, é uma linguagem visual para documentação de projetos e padrões de software, tudo tem representação gráfica. A UML pode ser aplicada em diferentes áreas, pode documentar e transmitir qualquer informação da organização para os processos de negócios de software empresarial. Pretende ser uma via comum para a documentação e expressão de relações, comportamentos e ideias de alto nível em uma notação que é fácil de aprender e eficiente para escrever. Neste material estaremos vendo o significado por trás de vários elementos da UML[8].

Evolução da UML

A UML se tornou padrão para modelagem de aplicações de software e está crescendo na modelagem de outros domínios. Suas origens recaem em três métodos distintos: o método Booch, por Grady Booch; a técnica de modelagem de objeto, de autoria de James Rumbaugh; e Objectory, de Ivair Jacobson, conhecido como os três amigos, que produziram o que veio a ser a primeira versão de UML, em 1994. Em 1997, UML foi aceita pelo *Object Management Group* (OMG) e editado como UML v1.1.

Desde então passou por diversas revisões e refinamentos, cada revisão tenta atingir problemas e fraquezas identificadas nas versões anteriores, levando a interessante expansão e contração da linguagem.

Administração Central

Cetec Capacitações

Entendendo a classificação conceitual da UML

É importante entender que UML é uma linguagem, significa que ela tem tanto sintaxe como semântica. Quando se modela um conceito em UML, existem regras determinando como os elementos podem ser agrupados e o que isso significa quando eles são organizados em determinada forma.

Podemos aplicar UML em:

- Projetos de Software
- Comunicação de processos de software ou negócios
- Documentação de detalhes sobre um sistema, processo ou organização existente.

Podemos considerar também:

- Setores bancários e de investimentos
- Planos de saúde
- Defesa
- Computação distribuída
- Sistemas embutidos
- Vendas e varejo e suprimentos.

O bloco básico de montagem de UML é um diagrama e existem vários tipos, alguns com propósitos muito específicos (diagrama de tempo) e alguns com usos mais genéricos (diagrama de classe) [6][7].

Administração Central

Cetec Capacitações

Entendendo a aplicação da UML na elaboração de software

Como UML cresceu a partir do domínio de desenvolvimento de software, não é surpresa que esse seja o domínio em que ela tenha maior aplicação. Quando aplicada a software, a UML procura fazer uma ponte sobre o vácuo entre a ideia original para uma parte de software e sua implementação. A UML produz uma via para documentar e discutir os requisitos e cada um dos níveis (diagrama de caso de uso), algumas vezes um novo conceito para projetistas. Existem diagramas para documentar as partes de software que produzem certos requisitos (diagrama de colaboração). Existem diagramas para documentar exatamente como aquelas partes do sistema produzem seus requisitos (diagrama de sequência e de gráfico de estado).

Os livros que descrevem as versões anteriores de UML fizeram questão de enfatizar que ela não era uma linguagem de programação visual. Entretanto a UML 2.0 altera um pouco essa regra. Uma das maiores motivações para passar da UML 1.5 para a UML 2.0 era a possibilidade de acrescentar a capacidade de os modeladores documentarem mais comportamentos do sistema e automação das ferramentas. Uma técnica relativamente nova, chamada Arquitetura Dirigida ao Modelo (MDA), oferece o poder de se desenvolver modelos executáveis em que as ferramentas podem se conectar e aumentar o nível de abstração acima das linguagens de programação tradicionais. A UML 2.0 é central para a realização da MDA.

Um modelo UML proporciona uma visão do sistema, frequentemente uma das muitas visões necessárias para montar ou documentar o sistema completo. Os usuários novatos em UML podem cair na armadilha de tentar modelar tudo no sistema usando um único diagrama e acabar perdendo informações críticas.

Para que você se torne proficiente em UML, deverá entender o que cada diagrama tem para oferecer e saber quando aplicá-lo. Muitas vezes você viverá a situação em que determinado conceito pode ser expresso por certo número de diagramas, escolha aquele que melhor expressará a informação para os usuários.

Administração Central

Cetec Capacitações

Intercambio de diagrama, infraestrutura e Superestrutura UML

Fisicamente, UML é um conjunto de especificações do OMG. A UML 2.0 é distribuída em quatro especificações de intercambio de diagrama, a infraestrutura UML, a Superestrutura UML e a Linguagem de Restrição de Objetos (OCL). Todas estão disponíveis no site do OMG (<http://www.omg.org>).

As especificações de intercâmbio de diagramas foram escritas para fornecer uma maneira de compartilhar modelos UML entre diferentes ferramentas de modelagem. As versões anteriores de UML definiam um esquema XML para a documentação dos elementos utilizados em um diagrama UML, mas não capturavam quaisquer informações sobre como o diagrama era montado. Para resolver esse problema, as especificações de intercambio de diagramas desenvolveram juntamente com o mapeamento de outro esquema XML para uma representação Gráfica de Vetor Escalável (SVG). As especificações de intercambio de diagramas tem sido usada somente por vendedores de ferramentas, embora o OMG faça um esforço para incluir “ferramentas de projetistas”. A infraestrutura UML define os conceitos fundamentais, de baixo nível, centrais, de maior profundidade em UML no geral, ela não é utilizada pelo usuário final, mas proporciona a base para superestrutura UML.

Administração Central

Cetec Capacitações

Cap. 2 – Modelagem Unificada UML

A modelagem é um meio de documentar ideias, relacionamentos, decisões e requisitos em uma notação bem definida que pode ser aplicada em diferentes áreas.

Em geral, um modelo UML é feito de um ou mais diagramas. Um diagrama representa, graficamente, elementos do mundo real e as relações entre elas. Esses elementos podem ser representações de objetos, construções de software ou uma descrição do computador de algum outro objeto.

É comum uma aplicação isolada aparecer em múltiplos diagramas, sendo que cada diagrama representa um interesse ou visão em particular da aplicação que está sendo modelada.

Diagramas estruturais e comportamentais

A UML divide os diagramas em duas categorias: **estruturais** e **comportamentais**.

Os diagramas **estruturais** podem ser utilizados para documentar a organização física, como um objeto se relaciona com outro.

Os diagramas **comportamentais** têm foco no comportamento dos elementos de um sistema. Por exemplo, para documentar requisitos, operações e alterações internas de estado para elementos.

Estruturais:

Diagrama de Classe: utilizam classes e interfaces para documentar detalhes sobre as entidades que formam o sistema e as relações estáticas entre elas. Os diagramas de classe estão entre os mais utilizados e podem variar em detalhes que podem ser depurados e aptos para gerar código fonte, a esboços rápidos em quadros brancos.

Administração Central

Cetec Capacitações

Diagrama de componente: mostram a organização e as dependências envolvidas na implementação do sistema. Eles podem agrupar elementos menores, como classes, em partes maiores, desdobráveis. A quantidade de detalhes que são utilizados em diagramas de componentes varia, dependendo do que se está tentando mostrar. Algumas pessoas simplesmente exibem a versão final, desdobrável de um sistema, outras, exibem a funcionalidade proporcionada por determinado componente e como ela a executa internamente.

Diagrama de estrutura compostas: são novos na UML 2.0. A medida que o sistema se torna mais complexo, as relações entre os elementos também se tornam mais complexas. Conceitualmente, os diagramas de estruturas compostas ligam os diagramas de classe e implementação que os componentes fazem. Ao invés disso, as estruturas compostas mostram como os elementos do sistema se combinam para formar padrões complexos.

Diagrama de desdobramento: mostram como o sistema é realmente executado e atribuído as várias partes de hardware. Você utilizara os diagramas de desdobramento normalmente para mostrar como os componentes funcionam durante a execução.

Diagrama de pacote: são realmente tipos especiais de diagramas de classe. Eles utilizam a mesma notação, mas o foco está em como as classes e interfaces são agrupadas.

Diagramas de Objeto: utilizam a mesma sintaxe que os diagramas de classes e mostram como as etapas reais de classe são relacionadas num instante específico. Você pode usar os diagramas de objetos para mostrar as relações dentro do sistema em momento específico.

Administração Central

Cetec Capacitações

Comportamentais:

Diagramas de comunicação: tipo de diagrama de interação que tem o foco sobre os elementos envolvidos num comportamento em particular e nas mensagens que eles passam adiante e para trás. Os diagramas de comunicação enfatizam mais os objetos envolvidos do que a ordem e natureza das mensagens trocadas.

Diagramas de interação resumidos: versões simplificadas de diagramas de atividade. Ao invés de enfatizar a atividade de cada passo os diagramas de interação resumidos enfatizam quais elementos estarão envolvidos no desempenho daquela atividade. As especificações UML descrevem diagramas de interação ao enfatizar quem tem o foco de controle durante a execução de um sistema.

Diagramas de sequência: tipo de diagrama de interação que enfatiza o tipo e a ordem das mensagens passadas entre os elementos durante a execução. Os diagramas de sequência são o tipo mais comum de diagramas de interação, além de ser muito intuitivos para os novos usuários de UML.

Diagrama de máquinas de estado: documentam as transições de estado internas de um elemento pode ser tão pequeno quanto uma classe simples ou tão grande quanto o sistema completo. Os diagramas de máquina de estado são normalmente utilizados para modelar sistema embutidos e especificações, ou implementações de protocolo.

Diagramas de tempo: tipo de diagrama de interação que enfatiza as especificações do detalhamento de tempo para mensagens. São frequentemente utilizados para modelar sistemas de tempo real, tais como comunicações por satélites ou entre partes de hardware. Eles têm notação específica para indicar quanto tempo o sistema tem para processar ou responder as mensagens, e como interrupções externas são tratadas na execução.

Diagramas de caso de uso: documentam os requisitos funcionais para um sistema. Eles fornecem uma visão independente de implementação e do papel que o sistema deve fazer, permitindo ao modelador manter o foco nas necessidades do usuário, ao invés de tê-los nos detalhes de realização[5] [3].

Administração Central

Cetec Capacitações

Classificando os diagramas

Para classificarmos qual o melhor diagrama para a implementação de um projeto, vamos aprender o conceito de visões de um sistema, esse conceito auxilia o modelador a encontrar diagramas que ajudam no transporte de informações correta, dependendo dos objetos. No geral, os modelos são frequentemente divididos entre o que se chama visões 4+1 do sistema. Esta notação representa quatro visões distintas de um sistema e um resumo de como tudo se acopla. As quatro visões são:

Visão de projeto: documenta as classes, interfaces e padrão que descrevem a representação do domínio de um problema e como o software será montado para resolve-lo. A visão do projeto quase sempre utiliza diagramas de classes, diagrama de objeto, diagramas de atividade, diagramas de estrutura compostas e diagramas de sequência para carregar o projeto de um sistema. A visão de projeto normalmente não encaminha a implementação ou execução do sistema.

Visão de desdobramento: documenta os meios com que o sistema está configurado, instalado e executado. Ela geralmente consiste de diagramas de componentes, diagramas de desdobramento e diagramas de interação. A visão de desdobramento documenta o meio com o qual o layout físico do hardware se comunica para executar o sistema e pode ser utilizada para exibir lista de falhas, redundâncias e topologia da rede.

Visão de implementação: enfatiza os componentes, arquivos e recursos usados por um sistema. Tipicamente a visão da implementação tem o foco nas gerências de configuração de um sistema, quais componentes dependem de quais arquivos fontes implementam quais classes, etc. As visões de implementação quase sempre usam um ou mais diagramas de componentes e podem incluir diagramas de interação, diagramas gráficos de estado e diagramas de estrutura composta.

Visão do processo: tem a finalidade de documentar informações sobre cooperação, desempenho e escalabilidade. As visões do processo normalmente utilizam alguma

Administração Central

Cetec Capacitações

forma de diagrama de interação e diagramas de atividades para mostrar como um sistema realmente se comporta em operação [3] [5].

As quatro visões distintas de um sistema são tomadas juntas pela visão final.

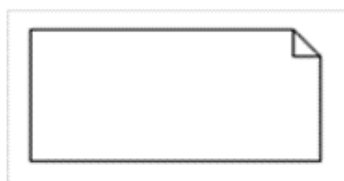
Visão de Caso de uso: documenta a funcionalidade requerida pelos usuários finais. O conceito de usuário final é deliberadamente amplo na visão de caso de uso, inclui os usuários básicos, o administrador do sistema os testadores e potencialmente os próprios desenvolvedores.

A visão de caso de uso é normalmente dividida em colaborações que ligam um caso a uma ou mais das quatro visões básicas. A visão de caso de uso inclui diagramas de caso de uso e, geralmente, usa vários diagramas de interação para mostrar detalhes do caso de uso.

Notas

UML fornece um elemento de notas para adicionar informações ao diagrama. O símbolo de nota é um retângulo com orelha dobrada no canto superior direito com uma linha pontilhada opcional para ligá-lo a algum elemento.

Figura 1 - Notas



Fonte: O autor.

Em geral, usamos as notas para documentar representar qualquer elemento em seu diagrama.

Administração Central

Cetec Capacitações

As notas são usadas para expressar informações adicionais que tem, ou não, a sua própria notação, ou que poderiam atravancar um diagrama se fossem anexadas diretamente ao elemento. Algumas ferramentas permitem a inclusão de links de URL nas notas, proporcionando uma maneira fácil para navegar de um diagrama ao próximo, ou até mesmo links em html.

As principais regras para modelagem

Por fornecer uma linguagem comum para documentação de informações sobre funcionalidade, a UML é deliberadamente aberta para permitir a flexibilidade necessária para modelar diferentes domínios. Existem algumas regras importantes ao usar UML:

- UML fornece uma linguagem para documentar informações que variam, dependendo do domínio do problema. É importante entender que não é necessário usar todas as partes de UML no modelo criado. Não é necessário usar todos os símbolos disponíveis a cada diagrama. Interessante usar apenas o que realmente ajuda a esclarecer a mensagem que está tentando apresentar e deixe de lado o que não for preciso.
- É comum a um modelo UML faltar alguns detalhes sobre o sistema. O truque é não deixar faltar detalhes-chave que poderiam impactar o projeto do seu sistema. Saber o que é um detalhe-chave, ao contrário de informações descartáveis, se consegue com a experiência. A diferença é o suporte de ferramentas que auxiliam em alterar o nível de abstração, dependendo das necessidades.
- A especificação de UML faz um bom serviço ao executar o trabalho duro de linguagem de modelagem. É importante que, dentro de uma organização ao grupo de usuários, possa ser estabelecido como e quando se deve usar um dispositivo de linguagem. Por exemplo, algumas organizações usam uma relação de agregação para iniciar um ponteiro em Java e uma relação de composição para indicar uma referência Java. Nada há de errado com essa distinção, mas ela não é imediatamente óbvia para alguém que não esteja

Administração Central

Cetec Capacitações

familiarizado com a técnica de modelagem. Criar um documento deve ajudar os usuários novatos a se familiarizarem rapidamente, além de auxiliar os experientes a pensarem como representar algo e a considerarem uma notação potencialmente melhor.

- UML inclui uma série de mecanismos que permitem a personalização e o refinamento da linguagem. Tais mecanismos como adornos, restrições e estereótipos proporcionam meios de documentação de detalhes específicos que não são facilmente expressos usando-se classificadores e relações. Geralmente eles são agrupados nos chamados perfis de UML [1] [3].

Cap. 3 Diagrama de Classe

Em UML, *diagramas de classes* são um de seis tipos de diagrama estrutural. Os diagramas de classe são fundamentais para o processo de modelagem de objetos e modelam a estrutura estática de um sistema. Dependendo da complexidade de um sistema, é possível utilizar um único diagrama de classe para modelar um sistema inteiro ou vários diagramas de classe para modelar os componentes de um sistema.

Os diagramas de classe são as cópias do sistema ou subsistema. Você pode utilizar os diagramas de classe para modelar os objetos que compõem o sistema, para exibir os relacionamentos entre os objetos e para descrever o que esses objetos fazem e os serviços que eles fornecem.

Os diagramas de classe são úteis em muitos estágios do design do sistema. No estágio de análise, um diagrama de classe pode ajudá-lo a compreender os requisitos do domínio do problema e a identificar seus componentes. Em um projeto de software orientado a objetos, os diagramas de classe criados durante os estágios iniciais do projeto contêm classes que normalmente são convertidas em classes e objetos de software reais quando você grava o código. Posteriormente, é possível refinar a análise e os modelos conceituais anteriores em diagramas de classe que mostrem as partes específicas do sistema, interfaces com o usuário, implementações lógicas e assim por

Administração Central

Cetec Capacitações

diante. Os diagramas de classe tornam-se, então, uma captura instantânea que descreve exatamente como o sistema funciona, os relacionamentos entre os componentes do sistema em vários níveis e como planejar implementar esses componentes.

Podemos utilizar diagramas de classe para visualizar, especificar e documentar recursos estruturais nos modelos. Por exemplo, durante as fases de análise e design do ciclo de desenvolvimento, é possível criar diagramas de classe para executar as seguintes funções:

- Capturar e definir a estrutura das classes e outros classificadores.
- Definir relacionamentos entre classes e classificadores.
- Ilustrar a estrutura de um modelo utilizando atributos, operações e sinais.
- Mostrar as funções e responsabilidades comuns do classificador que definem o comportamento do sistema.
- Mostrar as classes de implementação em um pacote.
- Mostrar a estrutura e o comportamento de uma ou mais classes.
- Mostrar uma hierarquia de herança entre classes e classificadores.
- Mostrar os trabalhadores e entidades como modelos de objetos de negócios.

Durante a fase de execução de um ciclo de desenvolvimento de software, é possível utilizar diagramas de classe para converter os modelos em código e converter o código em modelos.

Classes

Uma classe representa um grupo de informações que tem estado e comportamento comuns.

Podemos pensar em uma classe como uma cópia de um objeto em um sistema orientado a objetos. Falando-se em UML, uma classe é um tipo de classificador. Por exemplo Honda, Ford são todos carros, então podemos representa- los usando uma

Administração Central

Cetec Capacitações

classe chamada Carro. Cada modelo específico de carro é um exemplo daquela classe, ou um objeto. Uma classe pode representar um conceito tangível e concreto, tal como uma fatura, ela pode ser abstrata, tal como um documento ou veículo.

Podemos representar uma classe como uma caixa retangular dividida em compartimentos, um comportamento é simplesmente uma área de retângulo onde se escrevem as informações. O primeiro compartimento contém o nome da classe, o segundo, contém os atributos e o terceiro os métodos ou operações da classe.

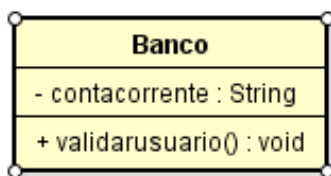
Podemos ocultar qualquer compartimento da classe, caso isso aumente a legibilidade do diagrama. Ao ler um diagrama, você não poderá assumir nada sobre o compartimento faltante, isso não significa que ele esteja vazio.

UML sugere que o nome da classe siga as seguintes recomendações:

- Comece com letra maiúscula
- Fique centrado no compartimento superior
- Seja escrito em negrito
- Seja em itálico, se for uma classe abstrata

Exemplo:

Figura 2 - Diagrama de Classe



Fonte: O autor

- A divisão superior exibe o nome da classe.
- A divisão do meio exibe uma lista de atributos.
- A divisão inferior exibe uma lista de operações.

Administração Central

Cetec Capacitações

É possível incluir divisões para exibir outros detalhes, como os sinais que as instâncias da classe podem receber[6] [8].

Objetos

Um objeto é a instancia de uma classe. Ou seja, a implementação das informações da classe. Por exemplo, podemos ter várias instâncias da classe carro, como: um carro vermelho de duas portas, um carro amarelo de quatro portas, cada instancia de carro é um objeto e pode receber um nome próprio, embora o mais comum é que sejam objetos ou diagramas de objeto sem nome, ou anônimos.

O padrão é mostrar o nome do objeto seguido por dois pontos e depois o tipo. Indica-se que isso é um exemplo de classe sublinhando o nome e o tipo.

Uma classe representa uma abstração de um conceito ou de uma coisa física, enquanto um objeto representa uma entidade concreta. Um objeto possui um limite bem definido e é significativo no aplicativo. Os objetos possuem as características listadas na tabela a seguir:

Tabela 1 - Características dos Objetos

Característica	Descrição
Estado	O estado é a condição na qual um objeto pode existir. O estado de um objeto é implementado com um conjunto de atributos e normalmente se altera com o passar do tempo.
Comportamento	O comportamento determina como um objeto responde a pedidos de outros objetos. O comportamento é implementado por um conjunto de operações.

Administração Central

Cetec Capacitações

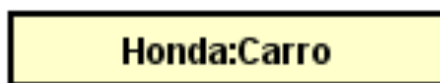
Identidade	A identidade de um objeto o torna exclusivo. Você pode utilizar a identidade exclusiva de um objeto para diferenciar entre múltiplas instâncias de uma classe se cada instância tiver o mesmo estado.
------------	---

Fonte: [1], [3].

Cada objeto deve possuir um nome exclusivo. Um nome completo de objeto possui três partes: nome do objeto, nome da função e nome da classe. Você pode utilizar qualquer combinação das partes quando nomear um objeto.

A imagem abaixo é um exemplo de objeto, da classe Carro, chamado Honda. Observe na figura 3 que os comportamentos foram ocultados.

Figura 3 - Objeto de Classe



Fonte: O autor.

Atributos

Os detalhes de uma classe (cor do carro, número da placa etc.) são representados como atributos. Os atributos podem ser tipos primitivos simples (inteiros, números de ponto decimal flutuante etc.) ou relações com outros objetos complexos.

Um atributo pode ser mostrado usando-se duas diferentes notações: alinhadas, ou por relações entre classes. Além disso, a notação está disponível para mostrar coisas como multiplicidade, singularidade e ordenamento.

Administração Central

Cetec Capacitações

Atributos Alinhados

Os atributos de uma classe podem ser listados direto na notação do retângulo. Tais notações são chamadas de atributos alinhados. Não há diferença semântica entre atributos alinhados e atributos por relação. Isso é simplesmente um problema de quantos detalhes você deseja apresentar (ou, no caso de primitivos como inteiros, quantos detalhes você pode apresentar).

Para representar um atributo dentro do corpo de uma classe, coloque o atributo no segundo compartimento de classe. UML se refere aos atributos alinhados como notação de atributos. Os atributos alinhados usam as seguintes notações:

Visibilidade / nome: tipo multiplicidade = padrão {sequência e restrição de propriedade} visibilidade = {+|-|#|~} multiplicidade ::= [inferior..superior]

Visibilidade

Indica a visibilidade do atributo. Utilize os símbolos +, -, #, ou ~ para, respectivamente, público, privado, protegido ou pacote.

/

Indica que o atributo é derivado. Um atributo derivado pode ser computado de outros atributos da classe.

Nome

É um substantivo, ou frase curta, nomeando o atributo. Normalmente a primeira letra é minúscula e a primeira letra de cada palavra subsequente é maiúscula.

Tipo

É o tipo de atributo como outro classificado, geralmente é a classe, interface ou um tipo de dados, como: int.

Administração Central

Cetec Capacitações

Multiplicidade

Especifica quantos exemplos de tipos de atributos são referenciados por esse atributo. Pode ser ausente (significando multiplicidade 1), um simples inteiro ou uma faixa específica de valores especificada entre colchetes por .. Utilize um * para representar o limite superior ou * por ele mesmo para significar zero ou mais.

Padrão

É o valor padrão do atributo

Sequência de prioridade

É uma coleção de propriedades, ou rótulos, que podem ser anexadas aos atributos. Elas são tipicamente especificadas do contexto e denotam coisas como ordenamento ou singularidade. Elas são limitadas por {} e separados por vírgulas

Restrições

Há uma ou mais restrições colocadas num atributo. Elas podem ser linguagem natural ou usar uma gramática formal tal como a OCL [2].

Atributos por Relação

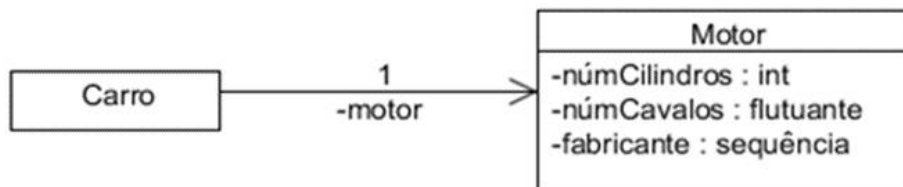
Você pode também representar atributos usando a notação de relação. Ela resulta num diagrama de classe, mas pode fornecer mais detalhes para os tipos de atributos complexos. A notação de relação também mostra exatamente como atributo este contido dentro de uma classe. Por exemplo, se você está modelando um Carro, pode mostrar que um carro contém um Motor muito mais claramente usando as relações do que você tentasse fazê-lo listando um atributo dentro do retângulo Carro. Entretanto, mostrar o nome do Carro por relação é provavelmente um desperdício porque é possível que seja só um texto.

Administração Central

Cetec Capacitações

Para representar um atributo usando relações, precisamos usar uma das relações de associação entre a classe que contém o atributo e a classe que representa o atributo, como pode ser visto na Figura 4. Ela mostra que as relações entre o carro e seu motor tem a multiplicidade de 1: um carro tem um motor:

Figura 4 - Relações



Fonte: O autor.

A notação de relação segue a mesma sintaxe da notação alinhada, embora o layout seja ligeiramente diferente. A visibilidade e o nome do atributo são colocados próximos a linha de relação. Não utilize colchetes para a multiplicidade, mas coloque a especificação da multiplicidade próxima ao classificador do atributo.

Assim como na multiplicidade, você pode colocar restrições nos atributos. Na notação de relação, escreva as restrições próximas ao classificador de atributo ao longo da linha de relação.

UML permite que a notação de relação expresse também as restrições entre atributos

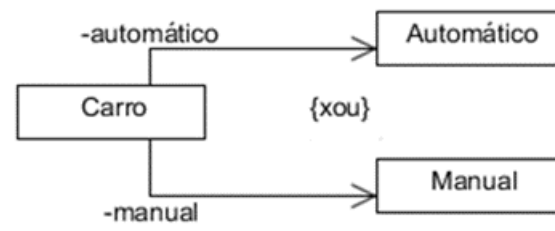
Assim como na multiplicidade, você pode colocar restrições nos atributos. Na notação de relação, escreva as restrições próximas ao classificador de atributo ao longo da linha de relação.

UML permite que a notação de relação expresse também as restrições entre atributos, como na figura 5:

Administração Central

Cetec Capacitações

Figura 5 - Restrições entre atributos



Fonte: O autor.

Na imagem anterior, a restrição UML padrão *xou* mostra que somente um automático ou manual, pode ser instalada em dado momento (ou exclusivo). Você precisa expressar essa restrição em uma nota se a notação de atributo alinhado foi usada.

Atributos Derivados

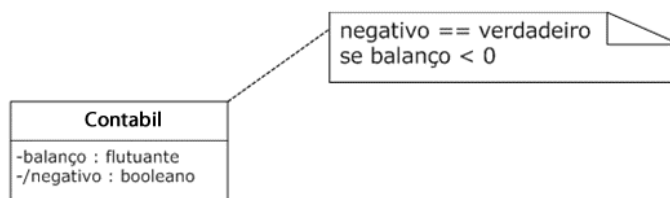
A notação derivada, representada pela barra inclinada / pode ser usada para indicar ao desenvolvedor que o atributo pode não ser estritamente necessário. Por exemplo, suponha que você modelou um controle contábil com uma classe simples chamada Contabil. Essa classe armazena o balanço atual como um número de ponto decimal flutuante, chamado balanço. Para informar se a conta está negativa, você adiciona um booleano chamado negativo. O fato de a conta estar negativa fica condicionado ao balanço estar positivo ou não e não no booleano acrescentado. Você pode indicar isso, para o desenvolvedor, mostrando que o negativo é um atributo derivado, com seu estado baseado no balanço.

A Figura 6 mostra como o balanço e o negativo podem ser representados usando uma nota que contém a relação.

Administração Central

Cetec Capacitações

Figura 6 - Nota que contém a relação



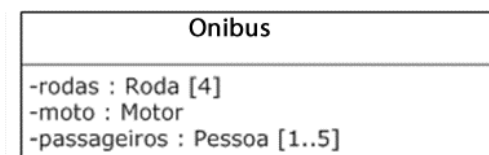
Fonte: O autor.

A especificação UML observa que um atributo derivado é tipicamente só de leitura, significando que o usuário não poderá modificar seu valor. Entretanto, se ao usuário for permitido modificar o valor, a classe deve atualizar a fonte da informação derivada adequadamente.

Multiplicidade de atributo

A multiplicidade de um atributo especifica quantos exemplos do tipo de atributo serão criados quando a classe a que ele pertence for instanciada. Para compreender, veja a Figura7: há colchetes na frente de alguns atributos e dentro deles um ou dois inteiros separados por dois pontos .. para representar a multiplicidade do atributo. Se o valor for especificado, fica implícito que é 1. Já um limite superior infinito pode ser representado por um *.

Figura 7 - Multiplicidade



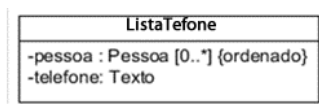
Fonte: O autor.

Administração Central

Cetec Capacitações

Quando um atributo tem multiplicidade maior que 1 pode ser especificado para ser ordenado. Se um atributo é ordenado, os elementos devem ser armazenados sequencialmente. Por exemplo, você pode especificar uma lista de nomes seja armazenada em ordem alfabética, fazendo com que a lista seja ordenada. Provisoriamente, os atributos não são ordenados. Para marcar um atributo como ordenado, especifique a prioridade ordenado depois do atributo, entre chaves, como mostra a figura 8:

Figura 8 - Multiplicidade maior que 1



Fonte: O autor.

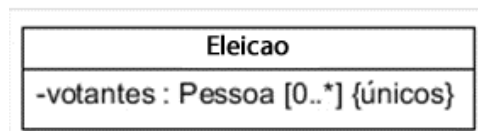
Singularidade

Além de ser ordenado, um atributo com multiplicidade maior que 1 pode ser requisitado para ser único. Se um atributo é requisitado para ser único, cada elemento desse atributo deve ser único. Provisoriamente, atributos com multiplicidade maior que 1 são únicos, significando que não pode haver duplicidades entre os elementos que esse atributo contém. Por exemplo, se uma classe Eleições contém uma lista de pessoas para votarem, cada pessoa só pode votar uma única vez, cada elemento da lista deverá ser único. Para fazer um atributo único, coloque a palavra-chave único depois do atributo, entre chaves, como mostra a Figura9. Para permitir que um atributo contenha duplicatas de um objeto, simplesmente use a propriedade *não único*.

Administração Central

Cetec Capacitações

Figura 9 - Permissão de Atributos



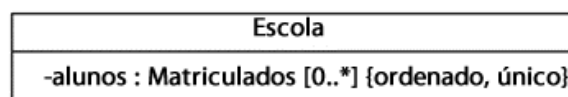
Fonte: O autor.

Tipos de Coleção

As especificações UML definem um conjunto de mapeamentos a partir das várias propriedades de ordenamento e de singularidade para os tipos de coleção UML.

Por exemplo, para mostrar que os alunos de determinada escola devam ser representados usando Conjunto Ordenado, você pode modelar o atributo alunos desta forma:

Figura 10 - Ordenado



Fonte: O autor.

Propriedades dos Atributos

Além das propriedades associadas à multiplicidade, um atributo pode ter um número de conjuntos de propriedades de forma a levar informações adicionais ao leitor do diagrama. As propriedades comuns definidas por UML são:

Administração Central

Cetec Capacitações

Somente Leitura

Especifica que o atributo não pode ser modificado depois que o valor inicial for definido. Essa propriedade mapeia para uma constante em uma linguagem de desenvolvimento. UML não especifica quando o valor inicial deve ser definido, mas ao especificar um valor padrão para um atributo ele é considerado o valor inicial e não pode ser alterado.

União

Especifica que o tipo de atributo é uma união dos possíveis valores para esse atributo. Frequentemente essa propriedade é utilizada com a propriedade derivada para indicar que um atributo é uma união derivada de outro conjunto de atributos.

Subconjuntos <nome-atributo>

Especifica que esse tipo de atributo é um subconjunto de todos os valores validos para o atributo dado. Essa não é uma propriedade comum, mas se utiliza, é associada com subclasses de um tipo de atributo.

Redefinir <nome-atributo>

Especifica que esse atributo opera como um nome adicional para o atributo dado. Embora incomum, esse atributo pode ser utilizado para mostrar que uma subclasse tem um atributo que é um nome adicional para um atributo de uma superclasse.

Administração Central

Cetec Capacitações

Composto

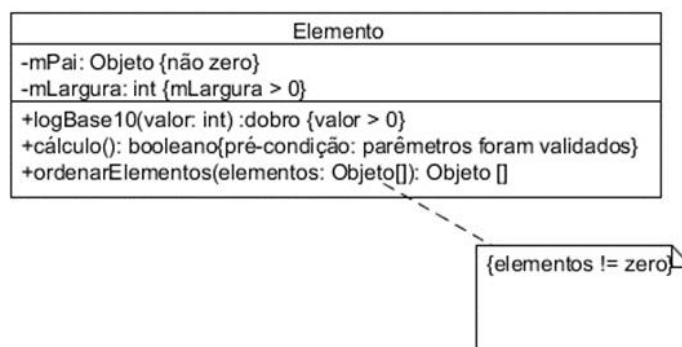
Especifica que esse atributo é parte de uma relação todo-parte com o classificador [2].

Restrições

As restrições representam limitações colocadas sobre um elemento. Elas podem ser de linguagem natural ou usar uma gramática formal tal como a OCL. Entretanto, elas devem avaliar uma expressão booleana. As restrições são basicamente exibidas entre chaves {} após o elemento que limitam, embora possam ser colocadas em uma nota e ligadas ao elemento usando-se uma linha tracejada.

Você pode nomear uma restrição especificando o nome seguido por dois pontos: antes da expressão booleana. Isto é feito frequentemente para identificar restrições em uma operação[1] [3] [5]. A Figura11 mostra várias restrições em atributos e operações:

Figura 11 - Restrições em Atributos



Fonte: O autor.

Administração Central

Cetec Capacitações

Atributos estáticos

Os atributos estáticos são atributos da classe que possuem o mesmo valor para todas as instâncias da classe. Os atributos são representados sublinhando a sua especificação, conforme mostrado na figura 12:

Figura 12 - Atributos

Único
<u>instância: Único</u>

Fonte: O autor.

Operações

Operações são as características das classes que especificam como chamar um comportamento particular, ou seja as ações que as classes executam. Exemplo, uma classe pode oferecer a operação de desenhar um retângulo na tela ou na contagem do número de itens selecionados em uma lista.

As operações devem ser colocadas em compartimento separado, com a seguinte sintaxe:

```
visibilidade nome (parâmetros) : tipo-retorno {propriedades}
```

Onde os parâmetros são escritos como:

```
nome_parâmetro direção : tipo [multiplicidade] = valor_padrão {propriedades}
```


Administração Central

Cetec Capacitações

Figura 13 - Elementos da Classe Porta

Porta
+obterTamanho(): Retângulo +definirTamanho(em nome: Sequência): nulo +obterComponentes(): Componente [0..*] #pintar(): nulo +definirVisibilidade(visível: booleano = verdadeiro): nulo -formatarTitulo(): nulo

Fonte: O autor.

Os elementos da sintaxe são:

Visibilidade

Indica a visibilidade da operação. Use os símbolos +, -, # ou ~, respectivamente, para público, privado, protegido ou pacote.

Nome

É uma frase curta para nomear a operação. As operações são geralmente expressões verbais que representam as ações que o classificador deve realizar em nome do chamador. A especificação UML recomenda que a primeira letra de uma operação seja minúscula, com todas as palavras seguintes começando com a letra Maiúscula e sendo emendadas.

Tipo-retorno

É o tipo de informação que a operação retorna, se se houver retorno. Se nenhuma informação é retornada da operação (chamado de sub-rotina, em algumas linguagens), o tipo de retorno deve ser nulo (void). Se a operação retorna um valor (chamado de

Administração Central

Cetec Capacitações

função, em algumas linguagens), você deve mostrar o tipo do valor retornado – como outro classificador, um tipo primitivo ou uma coleção. A especificação do tipo de retorno é opcional. Se não for especificado, não se pode assumir nada sobre o valor de retorno da operação.

Propriedades

Especifica as restrições e as propriedades associadas a operação. Elas são opcionais.

Os parâmetros dos elementos da sintaxe são:

Direção

Uma parte opcional da sintaxe que indica como um parâmetro é usado por uma operação: in (entrada), inout (entrada e saída), out (saída) ou return (retorno).

Nome_parâmetro

É um substantivo nomeado o parâmetro. Normalmente, o nome do parâmetro começa com letra minúscula, com todas as palavras subsequentes começando com letra Maiúscula.

Tipo

É o tipo de parâmetro. Ele pode ser outra classe, interface, coleção ou tipo primitivo.

Administração Central

Cetec Capacitações

Multiplicidade

Especifica quantas instancias de tipo do parâmetro estão presentes. Pode estar ausente (o que significa multiplicidade de 1), um único número inteiro, uma lista de inteiros separados por vírgulas ou uma sequência de valores inteiros indicados entre parênteses () e separados por dois pontos.. . Um limite infinito superior pode ser representado por um asterisco *.

Valor_padrão

Especifica o valor padrão do parâmetro. Ele é opcional e, se não estiver presente, não mostre o sinal de igual.

Propriedades

Qualquer propriedade relacionada aos parâmetros deve ser especificada entre chaves { }. As propriedades são definidas no contexto de um modelo específico, com algumas exceções: ordenados, somente leitura e único. Propriedades são opcionais para os parâmetros e, se não forem usadas, não mostre as chaves.

Restrições de Operações

Uma operação pode ter várias restrições e ela associadas, que ajudam a definir como a operação interage com o resto do sistema. Juntas, as restrições a uma operação, estabelecem o contrato ao qual a implementação deve obedecer.

Restrições em uma operação seguem a notação usual de uma restrição, e devem ser colocadas após a assinatura de operação ou em uma nota anexada[1] [3] [5].

Administração Central

Cetec Capacitações

Pré-Condições

As pré-condições são requisitos que o sistema deve ter antes de uma operação ser executada. Praticamente falando, não se pode expressar todo o estado do sistema e, ao invés disso, as pré-condições expressam os valores válidos para os parâmetros, o estado da classe proprietária da operação ou alguns atributos-chave do sistema.

A especificação afirma explicitamente que a operação não precisa checar as pré-condições da operação antes da execução. Teoricamente, a operação não vai sequer ser invocada quando as condições não são cumpridas. Na prática, poucas linguagens oferecem tal proteção. Se alguém tem tempo de expressá-las, terá interesse em verificar se as condições estão corretas na execução de uma operação.

A figura 14 mostra alguns exemplos de pré-condições:

Figura 14 - Processador de Pedidos

ProcessadorDePedidos
+calcularTotal(): flutuante {pré-condição: cart.itens.contagem > 0}
+despacharItens(destino: Endereço): booleano {pré-condição: pagamento foi verificado}

Fonte: O autor.

Pós-Condições

As pós-condição são requisitos a que o sistema deve atender após uma operação ser executada. Assim como as pré-condições, as pós-condições normalmente expressam o estado de um ou mais atributos-chave do sistema, ou alguma garantia sobre o estado da classe da operação.

Administração Central

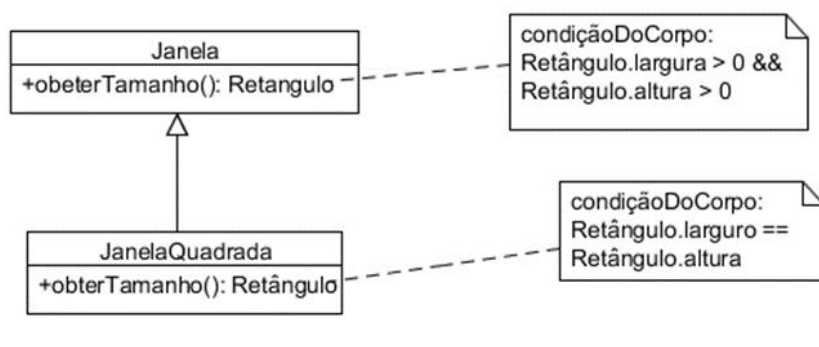
Cetec Capacitações

Condições do corpo

Determinada operação pode ter uma condição do corpo que restringe o valor de retorno. A condição do corpo é separada da pós-condição porque pode ser substituída por métodos de subclasses da classe que a contém. Por exemplo, uma classe chamada Janela pode especificar uma condição do corpo para um método chamado obterTamanho(), que requer que o comprimento e a largura de uma janela sejam diferentes de zero. Uma subclasse chamada JanelaQuadrada pode fornecer sua própria condição do corpo definido que a largura seja igual a altura. A condição do corpo é similar a pré e pós-condição do corpo definido que a largura seja igual a altura. A condição do corpo é similar a pré e pós-condição no que se refere ao fato de que a restrição deve ser expressa em linguagem natural ou em OCL

A figura 15 mostra um exemplo de condição do corpo em uma operação:

Figura 15 - Condição do Corpo



Fonte: O autor.

Operação Query

Uma operação pode ser declarada como operação de consulta (query), se a execução da operação não modificar a classe da operação. Na prática, usa-se a propriedade de

Administração Central

Cetec Capacitações

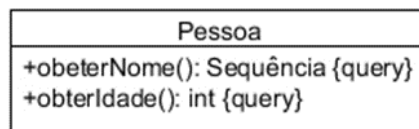
consulta para indicar determinado método que não altera nenhum atributo significativo do objeto.

O importante é que o estado do sistema, a partir de uma perspectiva externa, não seja alterado pelo método de consulta e que não haja efeitos colaterais para chamar o método.

O método de consulta é indicado colocando-se a restrição da consulta após a assinatura da operação. Por exemplo, uma operação denominada obterIdade(), que simplesmente retorna um inteiro sem alterar qualquer valor da classe, seria considerada um método de consulta. Em Java, isso é normalmente mapeado como um método de restrição.

Confira este exemplo da figura 16, de classe com método de consulta.

Figura 16 - Consulta



Fonte: O autor.

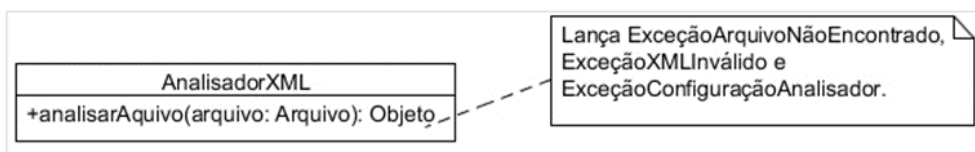
Exceções:

Embora não seja tecnicamente restrição para uma operação podem ser expressas usando-se uma notação semelhante. As exceções são tipicamente as outras classes (muitas vezes estereotipadas com a palavra-chave Exceção, embora isto seja só uma convenção), que são “invocadas” caso a operação gere um erro. As exceções podem ser listadas em nota anexada a operação, usando-se uma linha tracejada, conforme o Figura 17:

Administração Central

Cetec Capacitações

Figura 17 - Exceções



Fonte: O autor.

Métodos

Método é uma implementação da operação. Cada classe geralmente fornece uma implementação para as suas operações, ou as herdas de sua superclasse (a classe pai). Se a classe não fornece uma implementação para a operação e um método não é fornecido pela superclasse, a operação é considerada abstrata. Como os métodos são implementações das operações, não existe notação para um método, bastando mostrar a operação em uma classe.

Classes Abstratas

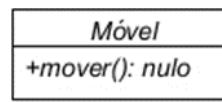
Classe abstrata é uma classe que normalmente oferece uma assinatura de operação, mas nenhuma implementação. No entanto, você pode ter uma classe abstrata que não possui operações. Uma classe abstrata é útil para identificar funcionalidades comuns em vários tipos de objetos. Por exemplo, uma classe abstrata chamada móvel: o objeto móvel tem uma posição atual e tem a capacidade de mudar para outro lugar, utilizando a operação denominada mover(). Pode haver várias especializações desta classe abstrata – um Carro e uma pessoa, sendo que cada uma delas fornece uma implementação diferente de mover(). Como a classe Móvel não tem implementação para mover(), ela é uma classe abstrata.

Indica-se que uma classe é abstrata, escrevendo seu nome, assim como o de cada operação, em *itálico*. Confira a figura 18.

Administração Central

Cetec Capacitações

Figura 18 - Classe Abstrata



Fonte: O autor.

A classe abstrata não pode ser instanciada, ou seja, sempre será superclasses terão que implementar todas as operações da classe abstrata e podem ser instanciadas.

Relações

As Classes, isoladamente, não fornecem informações de como um sistema é projetado. A UML oferece várias maneiras de representar as relações entre as classes. Cada tipo de relação UML representa um tipo diferente de relação entre classes, possuindo detalhes que não são totalmente captadas na especificação UML. Ao modelar com base no mundo real, você deve ter certeza que são totalmente captadas na especificação UML. Ao modelar com base no mundo real, você deve ter certeza que o público-alvo entendera que você está lidando com várias relações.

Dependência

A mais fraca relação entre classes é uma relação de dependência. A dependência entre classes significa que determinada classe usa ou tem conhecimento de outra classe. É tipicamente uma relação transiente, ou seja, a classe dependente interage brevemente com a classe alvo, mas normalmente não mantém relacionamento com ela durante todo o tempo.

As dependências são normalmente lidas como “...usa um ...”. Por exemplo, suponha que você tem a classe chamada Janela, que chama a classe chamada

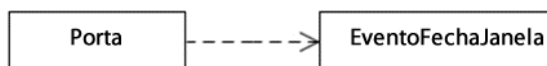
Administração Central

Cetec Capacitações

EventoFechaJanela quando ela está prestes a ser fechada, você pensaria “Janela usa um EventoFechaJanela”.

A dependência entre as classes é mostrada através de uma linha tracejada com uma seta apontando da classe dependente para a classe usada, conforme mostra a figura 19.

Figura 19 - Dependência



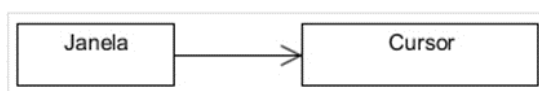
Fonte: O autor.

Associação

As associações são mais fortes do que as dependências e geralmente indicam que a classe mantém relacionamento com outra classe durante um período de tempo prolongado. O tempo de vida dos objetos não está ligado (ou seja, um pode ser destruído sem necessariamente destruir o outro).

As associações são normalmente lidas como “... tem um...”. Por exemplo, suponha que você tem a classe chamada Janela, que é uma referência para o cursor do mouse, você diria “Janela tem um Cursor”. Observe que há uma linha tênue entre “... tem um..” e “... possui um ...”. Neste caso, a janela não possui o Cursos, ele é compartilhado entre todas as aplicações do sistema. No entanto, Janela tem uma referência a ele, de forma que ela pode escondê-lo, mudar sua forma, etc. Você mostra a associação com uma linha sólida entre as classes participantes do relacionamento, conforme mostra a figura abaixo:

Figura 20 - Associação



Administração Central

Cetec Capacitações

Fonte: O autor.

Navegabilidade

Associação tem notação explícita para expressar navegabilidade. Se for possível navegar de uma classe a outra, mostre uma seta na direção da classe a qual se pode navegar. Se for possível navegar em ambos os sentidos, é prática comum não mostrar qualquer seta.

Mas a especificação UML define que, se você suprimir todas as setas, não conseguirá distinguir a associação não navegável da associação de dois sentidos. Como é extremamente raro usar a associação não navegável no mundo real, então é improvável que isso seja um problema.

É possível proibir a navegação de uma classe a outra. Para isso, coloque um **x**, na linha de associação, do lado que não se poderá navegar.

A imagem abaixo mostra uma associação entre a classe chamada **Janela** e a classe chamada **Cursor**.

Como não é possível navegar da instancia de cursos para a instancia de Janela, mostra-se a seta de navegabilidade com o **X** no local adequado.

Figura 21 - Navegabilidade X



Fonte: O autor.

Administração Central

Cetec Capacitações

Denominando uma Associação

Associações podem ser representadas por vários símbolos para acrescentar informações ao modelo. O mais simples é uma seta sólida mostrando a direção em que o usuário deve ler a associação. É comum incluir uma frase curta, juntamente com a seta, para fornecer algum contexto para a associação. A frase usada com a associação normalmente não é gerada durante a codificação, servindo apenas para fins de modelagem. Veja este exemplo:

Figura 22 - Associação



Fonte: O autor.

Multiplicidade

Como normalmente as associações representam relacionamentos, são muitas vezes usadas para indicar os atributos de uma classe. Como explicado anteriormente em “Atributos por Relação”, você pode expressar quantas instâncias da classe estão envolvidas em determinado relacionamento. Se você não especificar um valor será assumida a multiplicidade de 1. Para mostrar um valor diferente, basta especificar a multiplicidade da classe. Note que se utiliza a multiplicidade em uma associação. Podem ser utilizados colchetes em torno dos valores, conforme mostra a figura 23:

Administração Central

Cetec Capacitações

Figura 23 - Multiplicidade Associação



Fonte: O autor.

As propriedades relativas as multiplicidades também podem ser aplicadas as associações.

Agregação

A agregação é uma versão mais forte da associação. Ao contrário da associação, na agregação normalmente as classes tem o mesmo tempo de vida, ou seja, serão criadas e destruídas juntas. As agregações são geralmente lidas como “... possui um ...”. Por exemplo, suponha que você tivesse a classe Janela, que armazena o nome, tamanho e posição na classe Retângulo. Você diria que “Janela possui um Retângulo”. A classe Retângulo pode ser compartilhada com outras classes, mas Janela tem um relacionamento íntimo com ela.

A agregação é mostrada com um losango próximo a classe proprietária e uma linha sólida apontando para a classe possuída, conforme mostra a figura 24:

Figura 24 - Agregação



Fonte: O autor.

Administração Central

Cetec Capacitações

Tal como acontece com a relação de associação, a navegabilidade e a multiplicidade podem ser mostradas em uma linha de agregação.

Composição

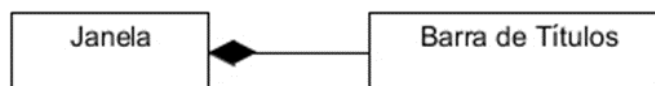
Composição representa a relação muito forte entre as classes, a ponto de contenção. Composição é usada para capturar um relacionamento todo-parte. A peça “parte” da relação pode ser envolvida em apenas uma relação de composição em um dado momento. O tempo de vida das instancias envolvidas no relacionamento de composição é quase sempre o mesmo. Se a instancia maior, possuidora, for destruída, quase sempre destrói a parte. UML permite que a parte seja associada a um novo proprietário antes da destruição, preservando assim a sua existência. Porém, esta é normalmente uma exceção e não a regra.

A relação de composição geralmente é lida como “... faz parte de...”, o que significa que se deve ler a composição da parte com o todo.

Por exemplo, para dizer que uma Janela em seu sistema deve ter uma barra de título, você pode representar isso com a classe chamada Barra de Títulos que “... é parte de ...” uma classe chamada Janela.

Esse relacionamento de composição deve ser mostrado com um losango preenchido junto à classe dominante e uma linha sólida apontando para a classe pertencente, conforme mostra a figura 25:

Figura 25 - Relacionamento de Composição



Fonte: O autor.

Administração Central

Cetec Capacitações

Tal como acontece com a relação de associação, a navegabilidade e multiplicidade podem ser mostradas em uma linha de composição.

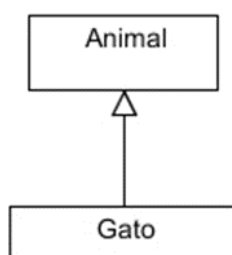
Generalização

Um relacionamento de generalização transmite que o objeto da relação é uma versão geral, ou menos específico, da classe-origem ou interface. As relações de generalização são frequentemente usadas para estabelecer afinidades entre diferentes classificadores. Por exemplo suponha que você tivesse a classe chamada Gato e a outra chamada Cachorro, a generalização para ambas seria a classe chamada Animal.

Generalizações são geralmente lidas como “... é um...”, a partir da classe mais específica e indo para a classe geral. Voltando ao exemplo do gato e do cão, você diria “um gato... é um ... Animal”.

O relacionamento de generalização deve ser mostrado por uma linha sólida com uma seta fechada apontando partir da classe específica para a classe geral, conforme mostra a figura 26:

Figura 26 - Relacionamento de Generalização



Fonte: O autor.

Ao contrário das associações, relacionamentos de generalização normalmente não são nomeados e não tem qualquer tipo de multiplicidade. UML permite herança múltipla, ou seja, uma classe pode ter mais de uma generalização com cada um, representando

Administração Central

Cetec Capacitações

um aspecto da classe ascendente. No entanto algumas línguas modernas (por exemplo, JAVA e C#) não suportam herança múltipla, usando interfaces ao invés disso.

Associação de Classes

Muitas vezes as relações entre dois elementos não é uma simples conexão estrutural. Por exemplo, um jogador de futebol pode ser associado a determinada liga de futebol por estar em certa equipe. Se a associação entre dois elementos for complexa, essa conexão pode ser representada usando-se uma classe de associação. A classe de associação possui nome e atributos, como uma classe normal.

Uma associação de classe deve ser mostrada como uma classe normal, com uma linha tracejada ligando-a a associação que representa.

Quando traduzido em código, a relação com classes de associação muitas vezes em três classes: uma para cada extremidade da associação e uma para a própria classe de associação. Pode haver ou não uma ligação direta entre os dois lados da associação e a implementação pode requerer que se acesse a classe de associação para chegar ao lado oposto da ligação. Em outras palavras, o jogador de futebol não pode ter uma referência direta para a Liga, mas ao invés disso pode ter uma referência para o Time. O Time teria, então, uma referência para a Liga. Como as relações serão construídas é uma questão de escolha de implementação. No entanto, os conceitos fundamentais de classe de associação mantêm-se inalterados.

Qualificadores de Associação

Relações entre elementos muitas vezes são introduzidos, ou indexados, por algum outro valor. Por exemplo, um cliente de banco pode ser identificado pelo número da

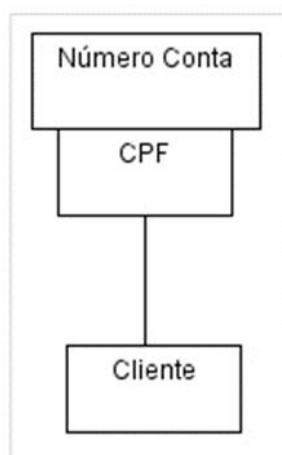
Administração Central

Cetec Capacitações

conta, ou pelo número de CPF. UML oferece qualificadores de Associação para capturar essas informações.

Um qualificador é tipicamente um atributo do elemento-alvo, embora isso não seja necessário. Um qualificador deve ser mostrado colocando-se um pequeno retângulo entre a associação e o elemento de origem. Desenhe o nome do qualificado (normalmente o nome de um atributo) no retângulo, conforme a figura 27:

Figura 27 - Qualificadores de Associação



Fonte: O autor.

Observe que a multiplicidade, entre o qualificado de associação e o Contribuinte, é 1.

Interfaces

Em modelagem UML, *interfaces* são elementos do modelo que definem conjuntos de operações que outros elementos do modelo, como classes ou componentes devem implementar. Um elemento de modelo de execução realiza uma interface pela substituição de cada um dos operandos que a interface declara.

Administração Central

Cetec Capacitações

É possível utilizar as interfaces em diagramas de classe e diagramas de componentes para especificar um contrato entre a interface e o classificador que realiza a interface. Cada interface especifica um conjunto de operações bem definido que possui visibilidade pública. As assinaturas da operação informam aos classificadores de execução que tipo de comportamento deve chamar, mas não como eles devem chamar esse comportamento. Muitos classificadores podem executar uma única interface, cada uma fornecendo uma execução exclusiva.

As interfaces suportam a ocultação de informações e a proteção de código do cliente pela declaração pública de determinados comportamentos ou serviços. As classes ou componentes que realizam as interfaces pela execução desse comportamento simplificam o desenvolvimento de aplicativos porque os desenvolvedores que gravam código do cliente precisam conhecer apenas as interfaces, não os detalhes da execução. Se você substituir classes, ou componentes que implementem interfaces, em seu modelo, não será necessário fazer um novo design do aplicativo se os novos elementos do modelo executarem as mesmas interfaces.

É possível especificar os seguintes tipos de interfaces:

- Interfaces fornecidas: essas interfaces descrevem os serviços que as instâncias de um classificador (fornecedor) oferecem a seus clientes
- Interfaces requeridas: essas interfaces especificam os serviços que um classificador necessita para executar suas funções e para cumprir suas próprias obrigações com seus clientes

Normalmente, uma interface possui um nome que reflete a função exercida em um aplicativo. Uma convenção comum é prefixar o nome da interface com uma barra para indicar que um elemento de modelo é uma interface.

Como a figura seguinte ilustra, o editor de diagramas exibe uma interface das seguintes maneiras:

Administração Central

Cetec Capacitações

Símbolo retangular de classe que contém a palavra-chave «interface». Essa notação também é chamada de visualização interna ou de classe.

Figura 28 - Notação de Interface



A segunda representação é a notação de bola e soquete. Esta representação mostra menos detalhes para a interface, mas é mais conveniente para mostrar as relações de classe. A interface é mostrada como uma bola com o nome da interface escrito abaixo. As classes dependentes dela são mostradas anexadas a um buraco correspondente da interface, conforme mostra a figura 29.

Figura 29 - Notação de Interface



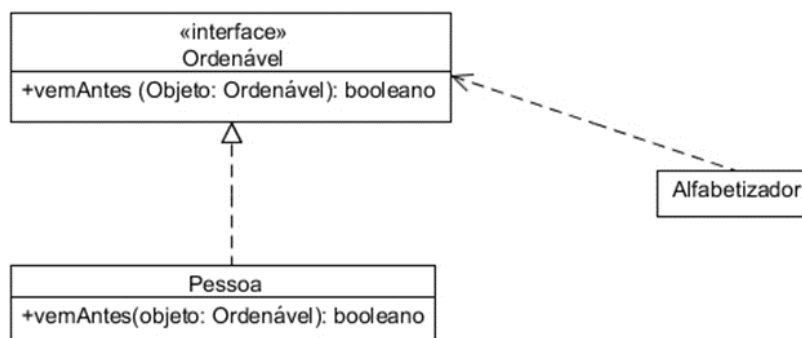
Fonte: O autor.

Você não pode instanciar diretamente uma interface. Ao invés disso, uma classe implementar uma interface e está fornece uma implementação para as operações e propriedades dela. Essa associação deve ser mostrada com uma linha tracejada da classe, levando a interface com uma ponta de flecha fechada ao final. As classes que são dependentes da interface são apresentadas por uma linha tracejada com uma seta aberta (dependência), conforme mostra a figura 30.

Administração Central

Cetec Capacitações

Figura 30 - Dependentes da Interface



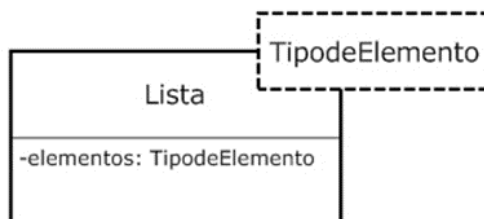
Fonte: O autor.

Classes Parametrizadas

Segundo Lima Cardoso e Araújo [6] a interface permite especificar como os objetos da classe irão interagir, a UML permite que sejam fornecidas as abstrações para o tipo de classe com a qual a classe pode interagir. Por exemplo, suponha que você irá escrever uma classe Lista que possa conter qualquer tipo de objeto. Para que a classe Lista seja capaz de suportar qualquer tipo de objeto, você irá especificá-los.

Podemos indicar que a classe é parametrizada desenhando um retângulo tracejado no canto superior direito da classe. Para cada tipo de elemento que você gostaria de especificar, irá escrever o nome do espaço reservado no retângulo, como mostra a figura 31:

Figura 31 - Classe Parametrizada



Fonte: O autor.

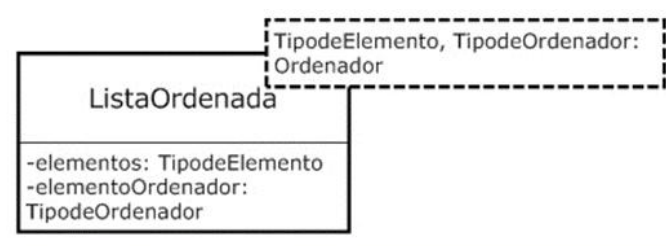
Administração Central

Cetec Capacitações

Obs.: Na prática usa-se apenas um T.

Você pode ter vários tipos parametrizados dentro de uma única classe, basta separar os nomes dos tipos com vírgula (,). Para restringir os tipos que o usuário pode substituir, indique isso com dois pontos (:) seguidos do nome do tipo. Confira a figura 32.

Figura 32 - Tipos Parametrizados



Fonte: O autor.

Especificar as restrições de um tipo é funcionalmente similar à especificação de uma interface, exceto que o usuário deve ser capaz de restringir ainda mais uma instancia de sua classe, indicando uma subclasse de seu tipo.

Quando o usuário cria uma instância de Lista, ela precisa especificar o tipo que será utilizado no lugar do tipodeElemento. Isso é chamado de tipo de ligação ao modelo. A ligação é mostrada com a palavra-chave <<amarrar>>, seguida da especificação de tipo utilizando a seguinte sintaxe:

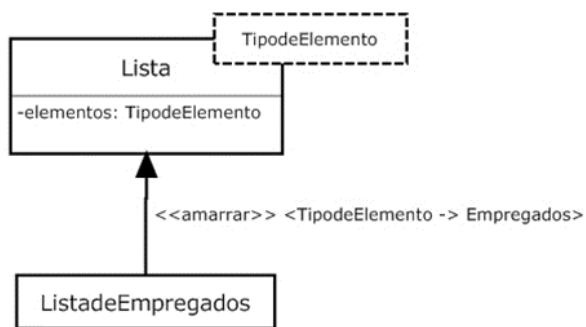
< TipoParametrizado -> TipoReal >

A sintaxe de ligação pode ser usada sempre que for referente a uma classe parametrizada para indicar que será usada uma versão vinculada dessa classe. Isso é chamado ligação explícita, como mostra o exemplo na figura 33:

Administração Central

Cetec Capacitações

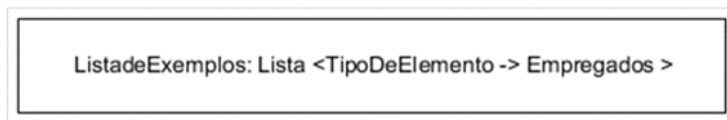
Figura 33 - Ligação Explícita



Fonte: O autor.

A palavra-chave <<amarrar>> também indica quais tipos devem ser usados com a instancia da classe parametrizada. Isso é chamado ligação implícita como mostra o exemplo a seguir:

Figura 34 - Ligação Explícita



Fonte: O autor.

Administração Central

Cetec Capacitações

Cap. 4 Diagramas de Sequência

Os diagramas de interação descrevem como grupos de objetos colaboram em algum comportamento. A UML define várias formas de diagrama de interação, das quais a mais comum é o diagrama de sequência.

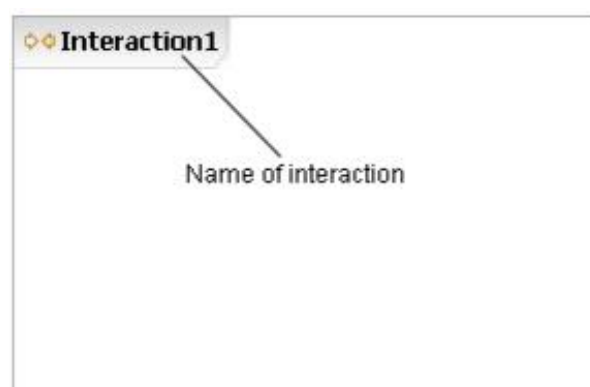
Normalmente, um diagrama de sequência captura o comportamento de um único cenário. O diagrama mostra vários exemplos de objetos e mensagens que são passadas entre esses objetos dentro de um caso de uso.

Quadros de Interação

Nos diagramas de sequência e de comunicação, um quadro de interação fornece um contexto ou limite para o diagrama no qual você cria elementos de diagramas, como linhas de vida ou mensagens, e no qual você observa comportamentos.

Conforme ilustrado na figura 35, um quadro de interação é exibido como uma área retangular com um rótulo de título ou uma figura descritora de pentágono, no canto superior esquerdo [2][6].

Figura 35 - Quadro de Interação



Fonte: Fowler; Lima Cardoso; Araujo [2][6].

Administração Central

Cetec Capacitações

O quadro e seu conteúdo representam uma interação em um diagrama de sequência ou comunicação. A etiqueta do título do quadro de interação é o nome da interação que o diagrama representa. Nos diagramas de sequência, um quadro pode representar fragmentos combinados, que representam construtores de cenário e usos de interação, que representam uma interação dentro de uma interface.

Linhas de Vida em Diagramas UML

Nos diagramas UML, como diagramas de sequência e comunicação, linhas de vida representam objetos que participam de uma interação. Por exemplo, em um cenário financeiro, linhas de vida podem representar objetos como um sistema bancário ou cliente. Cada instância em uma interação é representada por uma linha de vida.

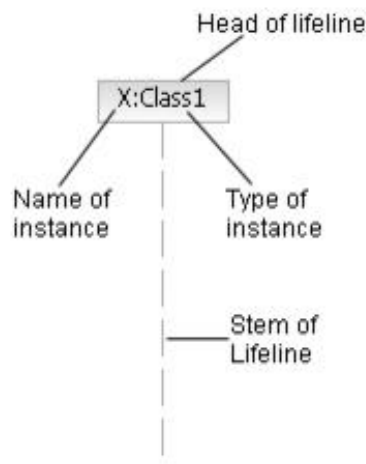
Linhas de Vida em Diagramas de Sequência

Como a seguinte figura ilustra, uma linha de vida em um diagrama de sequência é exibida com seu nome e tipo em um retângulo, que é chamado de cabeça. A cabeça está localizada no topo de uma linha tracejada vertical, chamada stem, que representa a linha de tempo da instância do objeto.

Administração Central

Cetec Capacitações

Figura 36 - Linha de Vida



Fonte: IBM [4].

Mensagens, que são enviadas e recebidas pela instância, aparecem na linha de vida na ordem de sequência. Você pode criar novas linhas de vida, criar linhas de vida de elementos existentes ou designar tipos de elementos para linhas de vida existentes.

Como a tabela a seguir, as linhas de vida podem indicar diversas ações nos diagramas de sequência.

Tabela 2 - Ações nos Diagramas de Sequência

Comportamento	Descrição
Criação	Você pode criar uma instância usando uma mensagem de criação. A mensagem de criação permite que um objeto crie novos objetos no diagrama de sequência.

Administração Central

Cetec Capacitações

Comunicação	Você indica mensagens com setas entre instâncias. A seta origina da linha de vida de origem que envia a mensagem e termina na linha de vida de destino que a recebe.
Execução	Uma especificação de execução mostra o comprimento de um comportamento de uma operação diretamente ou através de uma operação subordinada.
Destruição	Você pode destruir uma instância durante uma interação usando uma mensagem de destruição ou nó de parada. Uma mensagem de destruição é uma mensagem que termina na linha de vida de destino. Um nó de parada, representado por um X, marca o fim da linha de vida para indicar que ela terminou.

Fonte: IBM [4].

Linhas de Vida em Diagramas de Comunicação

Uma linha de vida em um diagrama de comunicação é representada por um retângulo que contém o nome e o tipo de instância.

customer2:Customer2

Administração Central

Cetec Capacitações

Mensagens em Diagramas UML

Uma mensagem é um elemento em um diagrama UML que define um tipo específico de comunicação entre instâncias em uma interação. Uma mensagem transporta informações de uma instância, que é representada por uma linha de vida, para uma outra instância em uma interação.

Tipos de Mensagens

Uma mensagem especifica um emissor e receptor e define o tipo de comunicação que ocorre entre linhas de vida. Por exemplo, uma comunicação pode chamar uma operação usando uma mensagem chamada síncrona ou uma mensagem de chamada assíncrona, pode emitir um sinal usando um sinal assíncrono e pode criar ou eliminar um participante.

É possível usar os cinco tipos de mensagens que estão listadas na tabela a seguir para mostrar a comunicação entre linhas de vida em uma interação.

Tabela 3 - Comunicação entre linhas de vida em uma interação

Tipo de mensagem	Descrição	Exemplo
Criar	Uma mensagem de criação representa a criação de uma instância em uma interação. A mensagem de criação é representada pela palavra-chave «create». A linha de vida de destino começa no ponto da	Em um cenário financeiro, um gerente de banco pode iniciar uma verificação de crédito para um cliente, enviando uma mensagem de criação para o

Administração Central

Cetec Capacitações

	mensagem de criação.	servidor.
Destruir	Uma mensagem de destruição representa a destruição de uma instância em uma interação. A mensagem de destruição é representada pela palavra-chave «destroy». A linha de vida de destino termina no ponto da mensagem de destruição e é indicada por um X.	Um gerente de banco, depois de iniciar uma verificação de crédito, pode fechar ou destruir o aplicativo de programa de crédito para um cliente.
Chamada síncrona	As chamadas síncronas, que estão associadas a uma operação, possuem uma mensagem de envio e uma de recebimento. Uma mensagem é enviada da linha de vida de origem para a linha de vida de destino. A linha de vida de origem é bloqueada de outras operações até receber uma resposta da linha de vida de destino.	Um caixa de banco pode enviar uma solicitação de aprovação de crédito para o gerente de banco e deve aguardar uma resposta antes de continuar o atendimento ao cliente.
Chamada assíncrona	As chamadas assíncronas, que estão associadas a operações, geralmente têm apenas uma mensagem de envio, mas também podem ter uma mensagem de resposta. Em	Um cliente bancário pode solicitar crédito, mas pode receber informações financeiras por telefone ou solicitar dinheiro a partir de um

Administração Central

Cetec Capacitações

	<p>contraste, para uma mensagem síncrona, a linha de vida de origem não é bloqueada do recebimento ou envio de outras mensagens. Você também pode mover os pontos de envio e recebimento individualmente para atrasar o tempo entre os eventos de envio e recebimento. Você pode optar por fazer isso se uma resposta não for sensível ao tempo ou sensível à ordem.</p>	<p>ATM, enquanto aguarda resposta da requisição de crédito.</p>
<p>Sinal assíncrono</p>	<p>As mensagens de sinal assíncrono estão associadas a sinais. Um sinal difere de uma mensagem porque nenhuma operação está associada ao sinal. Um sinal pode representar uma condição de interrupção ou condição de erro. Para especificar um sinal, você cria uma mensagem de chamada assíncrona e altera o tipo na visualização de propriedades da mensagem.</p>	<p>Uma agência de crédito pode enviar uma mensagem de sinal de erro ao gerente de banco que declara uma falha de conexão com o departamento de crédito.</p>
<p>Achados e perdidos</p>	<p>Uma mensagem perdida é uma mensagem que tem um remetente conhecido, mas o</p>	<p>Um agente externo envia uma mensagem para um gerenciador de banco. O</p>

Administração Central

Cetec Capacitações

	destinatário não é conhecido. Uma mensagem localizada é uma mensagem que não tem um remetente conhecido, mas tem um destinatário.	agente está fora do escopo do diagrama de sequência e, portanto, é uma mensagem localizada. Uma mensagem perdida pode ocorrer quando uma mensagem é enviada para um elemento fora do escopo do diagrama UML.
--	---	--

Fonte: IBM [4].

Uma mensagem assíncrona é o único tipo de mensagem para o qual é possível mover individualmente os pontos de envio e de recebimento. Você pode mover os pontos de uma mensagem assíncrona para manipular o atraso entre o evento de envio e de recebimento; o resultado é chamado de mensagem desviada. É possível criar uma mensagem assíncrona com ou sem uma especificação de execução de comportamento.

Uma mensagem auto-direcionada é uma mensagem enviada da linha de vida de origem para ela mesma. Uma mensagem auto-direcionada pode ser uma chamada recursiva ou uma chamada para outra operação ou sinal que pertence ao mesmo objeto.

A mensagem que a linha de vida de origem envia para a linha de origem de destino representa uma operação ou um sinal que a linha de vida implementa. É possível nomear e ordenar mensagens. A aparência da linha ou da cabeça de seta reflete as propriedades da mensagem.

Uma mensagem representa uma chamada de operação ou o envio e o recebimento de um sinal. Quando a mensagem representa uma operação, o nome da operação identifica a mensagem. Os argumentos da mensagem são transmitidos para a origem

Administração Central

Cetec Capacitações

do destino. A mensagem de retorno contém os argumentos da chamada de operação resultante. Quando uma mensagem representa um sinal, os argumentos da mensagem são o próprio sinal. Se a mensagem for uma chamada síncrona, uma mensagem de retorno ocorrerá a partir da linha de vida chamada para a linha de vida que chama, antes que a linha de vida que chama possa prosseguir.

Podemos identificar as mensagens, utilizando um nome ou uma assinatura de operação. Um nome identifica somente o nome da mensagem que não está associado com uma operação. Quando uma operação está associada a uma mensagem, o nome da operação substitui o nome. Uma assinatura de operação é exibida para identificar o nome da operação. Você pode usar assinaturas em diagramas durante a fase de design para fornecer detalhes para os desenvolvedores que codificam o design.

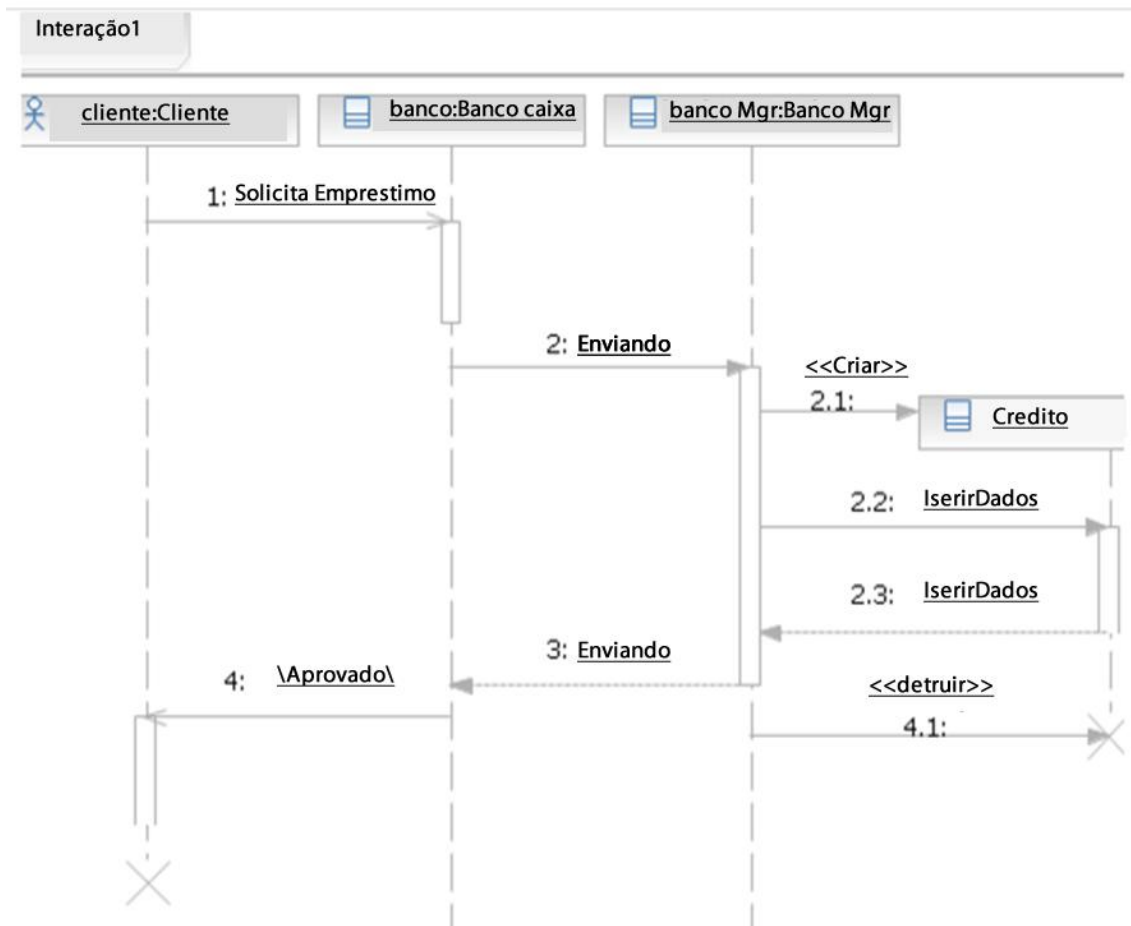
Como a figura 37 ilustra, mensagens são exibidas como uma linha com uma seta apontando na direção na qual a mensagem é enviada; ou seja, do fim da mensagem de envio para o fim da mensagem de recebimento. O exemplo a seguir mostra como as mensagens são exibidas em um diagrama de sequência que representa um cenário financeiro no qual um cliente bancário solicita um empréstimo seguindo esse processo.

- Um cliente fornece uma requisição de empréstimo para um caixa de banco.
- O caixa de banco envia a requisição para o gerente de banco para ser processada e aguarda a conclusão do gerente.
- O gerente do banco inicia a verificação de crédito, insere os dados e aguarda pelo envio de resultados pela agência de crédito.
- O gerente de banco recebe uma resposta da agência de crédito e envia uma mensagem para o caixa de banco que informa a decisão.
- O caixa de banco envia uma mensagem para o cliente que informa se o empréstimo foi aprovado.
- O gerente do banco fecha o programa da agência de crédito e o cliente completa a transação.

Administração Central

Cetec Capacitações

Figura 37 - Mensagens em um diagrama de Sequência



Fonte: IBM [4].

Administração Central

Cetec Capacitações

Fragmentos Combinados nos Diagramas de Sequência

Nos diagramas de sequência, fragmentos combinados são agrupamentos lógicos representados por um retângulo, que contém as estruturas condicionais que afetam o fluxo de mensagens. Um fragmento combinado contém operandos de interação e é definido pelo operador de interação.

O tipo de fragmento combinado é determinado pelo operador de interação. Você pode utilizar fragmentos combinados para descrever vários controles e estruturas lógicas de um modo compacto e conciso. O operador de interação identifica o tipo de instrução lógica ou condicional que define o comportamento de fragmento combinado.

Um fragmento combinado também pode conter fragmentos combinados aninhados ou os usos de interação contendo estruturas condicionais adicionais que representam estruturas mais complexas que afetam o fluxo de mensagens.

Usos da Interação nos Diagramas de Sequência

Nos diagramas de sequência, os usos de interação permitem que você faça referência a outras interações existentes. Você pode criar uma sequência completa e complexa de interações mais simples e menores.

Um uso de interação é representado por um quadro, semelhante a um fragmento combinado. A tag "ref" é colocada dentro do quadro e o nome da interação que está sendo referida é colocado dentro da área de conteúdo do quadro, junto com parâmetros e tipos de retorno.

Para começar a discussão, considerarei um cenário simples. Temos um pedido e vamos executar um comando sobre ele para calcular seu preço. Para fazer isso, o pedido precisa examinar todos os itens de linha nele presentes e determinar seus preços, os

Administração Central

Cetec Capacitações

quais são baseados nas regras de composição de preços dos produtos da linha do pedido. Tendo feito isso para todos os itens de linha, o pedido precisa então calcular um desconto global, que é baseado nas regras vinculadas ao cliente.

A seguir temos um diagrama de sequência que mostra uma implementação desse cenário. Os diagramas de sequência mostram a interação, exibindo cada participante com uma linha de vida, que corre verticalmente na página, e a ordem das mensagens, lendo a página de cima para baixo.

Uma das coisas interessantes a respeito de um diagrama de sequência é que quase não é preciso explicar a notação. Você pode ver que uma instância do pedido e na via mensagens obterQuantidade e obterProduto para a linha do pedido. Você também pode ver como mostramos o pedido chamando um método dele mesmo e como esse método envia obterInformaçãodeDesconto para uma instância de cliente.

O diagrama, entretanto, não mostra tudo muito bem. A sequência de mensagens obterQuantidade, obterProduto, obterDetalhesdoPreço e calcularInformaçãodeDesconto precisa ser feita para cada linha de pedido no pedido, enquanto calcularDesconto é chamado apenas uma vez. Você não pode saber isso a partir desse diagrama, embora apresentemos mais alguma notação para tratar disso, posteriormente.

A maior parte das vezes, você pode considerar os participantes de um diagrama de interação como objetos, conforme eles eram, na realidade, na UML 1. Mas, na UML 2, seus papéis são muito mais complicados e, explicar isso completamente está fora dos objetivos deste livro. Assim, uso o termo participantes, uma palavra que não é utilizada formalmente na especificação da UML. Na UML 1, os participantes eram objetos e, assim, seus nomes eram sublinhados; mas, na UML 2, eles devem ser mostrados sem o sublinhado, conforme fizemos aqui.

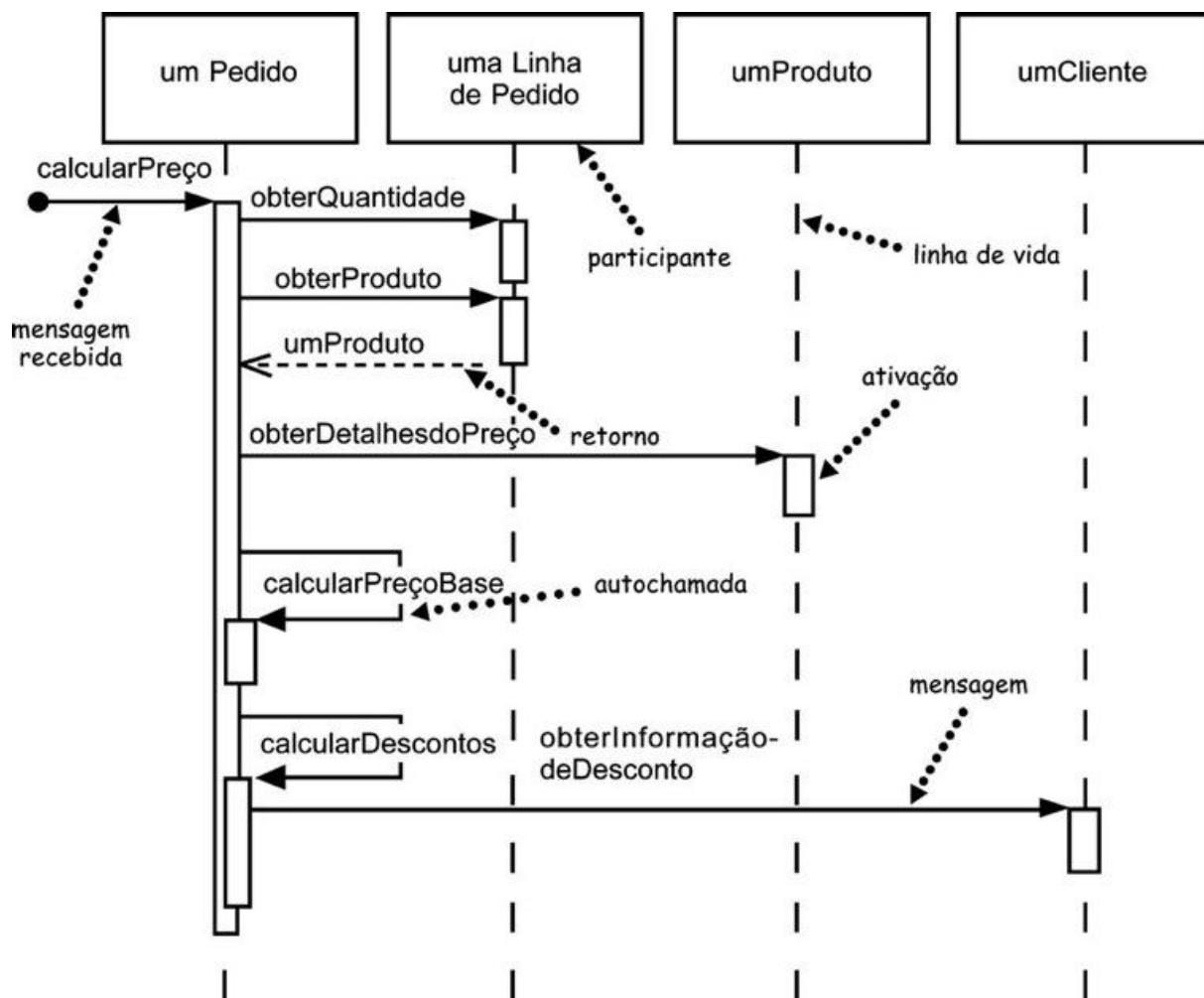
Nesses diagramas, dei nome aos participantes utilizando o estilo umPedido. Isso funciona bem na maior parte das vezes. Uma sintaxe mais completa é nome:

Administração Central

Cetec Capacitações

Classe, onde o nome e a classe são opcionais, mas você deve manter os dois-pontos, se usar a classe.

Figura 38 - Diagrama de Sequência



Fonte: O autor.

Administração Central

Cetec Capacitações

Cada linha de vida tem uma barra de ativação que mostra quando o participante está ativo na interação. Isso corresponde aos métodos de um dos participantes entrando na pilha. As barras de ativação são opcionais na UML, mas as considero extremamente valiosas no esclarecimento do comportamento. A única exceção é ao explorar um projeto durante uma sessão de projeto, pois elas são incômodas de desenhar em quadros brancos.

Frequentemente, a atribuição de nomes é útil para correlacionar participantes no diagrama. A chamada obterProduto aparece retornando umProduto, o qual tem o mesmo nome e, portanto, é o mesmo participante, que o objeto umProduto para o qual a chamada de obterDetalhesdoPreço é enviada. Note que usei uma seta de retorno apenas para essa chamada; fiz isso para mostrar a correspondência. Algumas pessoas utilizam retornos para todas as chamadas, mas prefiro usá-los somente onde eles acrescentam informações; caso contrário, eles apenas congestionam o diagrama. Mesmo nesse caso, você provavelmente poderia omitir o retorno, sem confundir seu leitor.

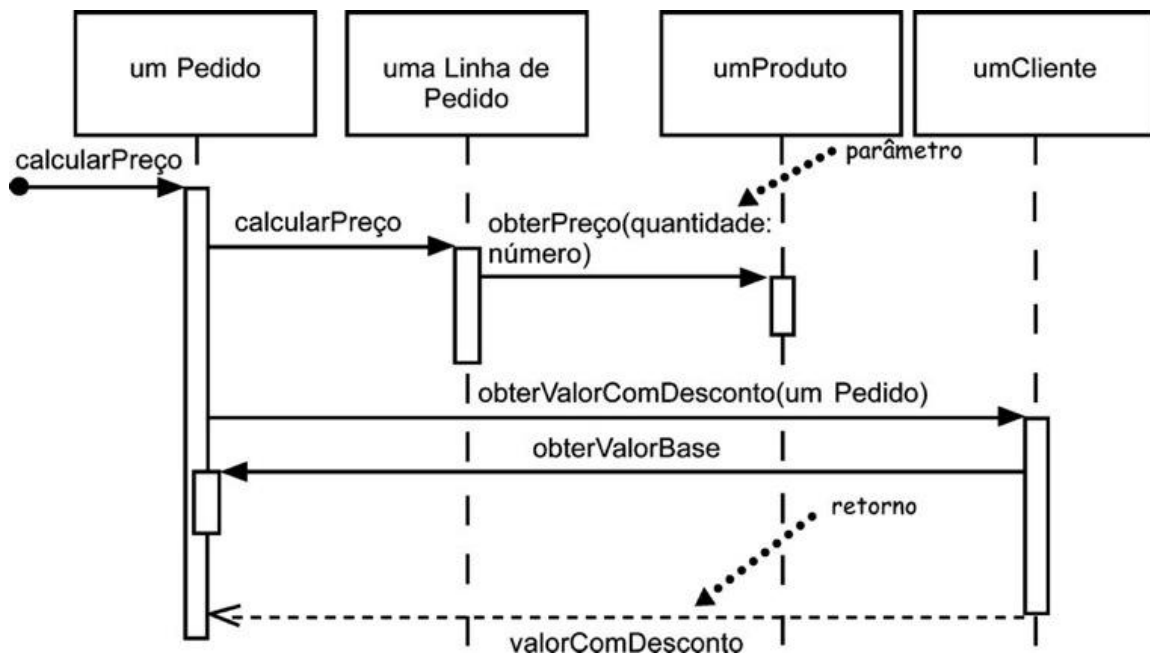
A primeira mensagem não tem um participante que a enviou, pois ela é proveniente de uma fonte indeterminada. Ela é chamada de mensagem recebida.

Para outra abordagem desse cenário, dê uma olhada na próxima figura. O problema básico ainda é o mesmo, mas a maneira como os participantes colaboram para implementá-lo é muito diferente. O Pedido pede para que cada Linha de Pedido calcule seu próprio Preço. A própria Linha de Pedido transmite o cálculo para o Produto; observe como mostramos a passagem de um parâmetro. Analogamente, para calcular o desconto, o Pedido chama um método do Cliente. Como ele precisa de informações do Pedido para fazer isso, o Cliente faz uma chamada reentrante (obterValorBase) no Pedido para obter os dados.

Administração Central

Cetec Capacitações

Figura 39 - Diagrama de Sequência



Fonte: O autor.

Administração Central

Cetec Capacitações

Cap. 5 Diagramas de Pacote

As classes representam a forma básica de estruturação de um sistema orientado a objetos. Embora elas sejam maravilhosamente úteis, você precisa de algo mais para estruturar sistemas grandes, os quais podem ter centenas de classes.

Os pacotes oferecem uma maneira de agrupar elementos UML relacionados e de fazer o escopo de seus nomes como, por exemplo, colocar todos os elementos que tem a ver com a renderização em 3D em um pacote chamado 3DGraphics. Diagramas de pacote fornecem uma ótima maneira de visualizar as dependências entre as partes de seu sistema e são usadas frequentemente para procurar problemas ou determinar a ordem de compilação.

Um pacote é uma construção de agrupamento que permite a você pegar qualquer construção na UML e agrupar seus elementos em unidades de nível mais alto. Seu uso mais comum é o agrupamento de classes e é dessa maneira que o estou descrevendo aqui, mas lembre-se de que você também pode usar pacotes para todos os outros elementos da UML.

Quase todos os elementos UML podem ser agrupados em pacotes, que incluem pacotes próprios. Cada pacote tem um nome que faz o escopo de todos os seus elementos. Por exemplo, suponha que você tivesse chamada Temporizador em um pacote chamado Utilidades, o nome qualificado para a classe seria Utilidades: Temporizador. Elementos do mesmo pacote podem se referir entre si, sem usar qualificador em seus nomes.

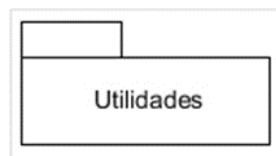
Representação

Para mostrar um pacote use um retângulo com uma aba anexada ao canto superior esquerdo, como na figura 40:

Administração Central

Cetec Capacitações

Figura 40 - Pacote

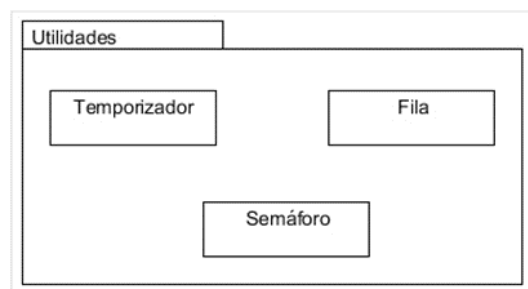


Fonte: O autor.

Os elementos contidos dentro de um pacote podem ser exibidos de duas maneiras diferentes:

- 1) Podem ser colocados dentro do retângulo grande – ao usar tal representação, escreva o nome do pacote na aba, como mostra a figura 41:

Figura 41 – Representação de Pacote



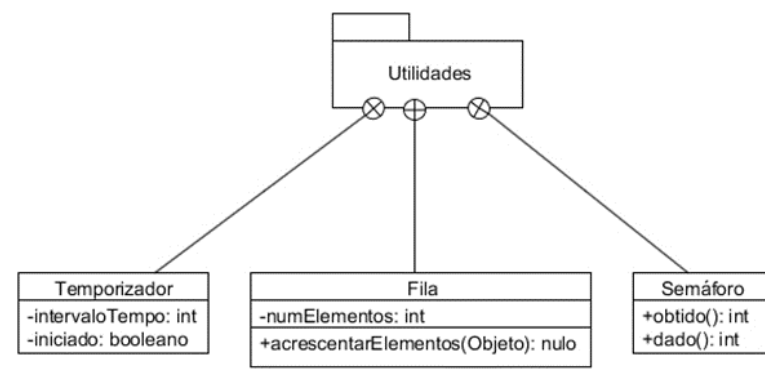
Fonte: O autor.

- 2) Use uma linha sólida, a partir do pacote, apontando para cada elemento contido. Coloque um círculo com um sinal de + no final da linha próximo ao pacote, para indicar confinamento. Ao usar esta representação, deverá mostrar o nome do pacote no retângulo grande, ao invés de na aba. Esta notação permite-lhe mostrar mais detalhes dos elementos do pacote, conforme indica a imagem:

Administração Central

Cetec Capacitações

Figura 42 - Representação de Pacote



Fonte: O autor.

Você não precisa mostrar todos os elementos contidos em um pacote, se não houver elementos apresentados, não podem ser feitas suposições sobre o que contém no pacote.

Visibilidade

Um pacote pode especificar as informações de visibilidade para elementos contidos e importados, porem os elementos podem ter apenas um dos dois níveis de visibilidade pública ou privado. Visibilidade publica significa que o elemento pode ser utilizado fora do pacote. Visibilidade privadas significa que o elemento só pode ser usado por outros elementos do mesmo pacote. Visibilidade privada é útil para marcar classes que representam um subsistema ou componente que não se deseja expor ao resto do sistema.

A visibilidade pública é mostrada colocando-se um sinal de adição + antes do nome do elemento. A visibilidade privada é mostrada utilizando- se um sinal de menos - .

Administração Central

Cetec Capacitações

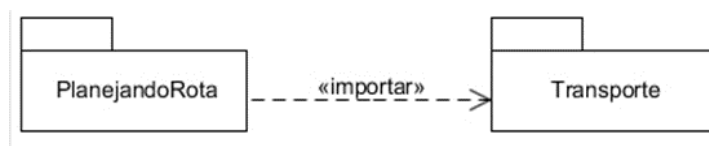
Importando e Acessando Pacotes

Para acessar elementos em um pacote, de outro pacote diferente, deve-se qualificar o nome do elemento que se está acessando. Por exemplo, se Carro é uma classe no pacote Transporte e você está tentando acessar a partir de outro pacote chamado PlanejandoRota, será preciso qualificar Carro como Transporte: Carro

Para simplificar o acesso a elementos de pacotes diferentes, UML permite que um pacote importe outra. Os elementos do pacote importado ficam disponíveis, sem qualificação, no pacote importador. Assim, se o pacote PlanejandoRota importou o pacote Transporte, você pode se referir a Carro sem qualquer qualificação dentro do pacote PlanejandoRota.

Para mostrar um pacote de importação deve-se desenhar uma linha tracejada com uma seta aberta a partir do pacote de importação para o pacote importado. Rotule essa linha com a palavra-chave <<importar>>, como mostra a figura 43:

Figura 43 - Pacote de Importação



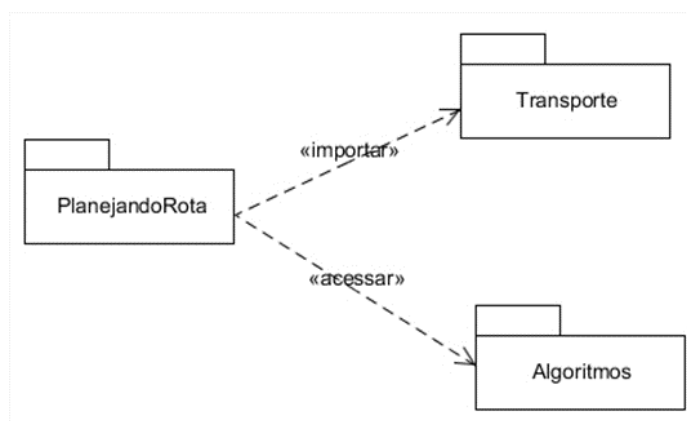
Fonte: O autor.

Por padrão, elementos importados recebem visibilidade pública no pacote importador. UML permite especificar se os elementos importados devem receber visibilidade privada, o que significa que não podem ser usados fora do pacote importador (incluindo todos os pacotes que podem importar este pacote). Para especificar a visibilidade privada, deve ser usada a palavra-chave <<acessar>>, ao invés da palavra-chave <<importar>>. Veja a figura 44:

Administração Central

Cetec Capacitações

Figura 44 - Pacotes



Fonte: O autor.

A importação e o acesso dos pacotes variam drasticamente no momento da implementação, dependendo da linguagem-alvo. Por exemplo, C# e Java tem um conceito explícito de pacotes e dos elementos de importação. Já C++ tem um conceito um pouco mais sutil de pacotes, chamado namespaces. Como os pacotes mapeiam a implementação, tudo dependerá do implementador.

Mixagem de Pacotes

UML contém um conceito um tanto complexo de mixagem de pacotes. Mixagem de pacotes varia de importação de pacotes, no fato de mixar. Por definição, cria relações entre classes de mesmo nome. A motivação por trás da mixagem de pacotes vem diretamente da evolução da UML.

A UML 2.0 define o conceito básico de elementos e permite que tipos de diagramas específicos possam estender um conceito básico sem a necessidade de fornecer um novo nome para ele. Por exemplo, UML estende vários conceitos de Máquina de

Administração Central

Cetec Capacitações

Estado Comportamental em conceitos Maquinas de Estado de Protocolo, mantendo seus nomes originais.

Quando um pacote é mixado com outra, qualquer classe do mesmo tipo e nome automaticamente estende (ou tem relação de generalização) a classe original.

A mixagem de pacotes é mostrada pela linha tracejada com a seta aberta, a partir do pacote mixador para o pacote resultante da mixagem. Rotule a linha com a palavra-chave <<mixar>>.

Figura 45 - Mixar



Fonte: O autor.

As regras para mixagem de pacotes são:

Membros privados de um pacote não são mixados.

As classes dos pacotes que executam a mixagem e tenham o mesmo nome e tipo de classes, obtém uma relação de generalização com as classes mixadas. Note que isso pode resultar em herança múltipla e UML permite isso.

É possível fazer referência a qualquer classe do pacote original, explicitando o escopo da classe e usando o nome do pacote original

As classes que só existem no pacote mixado, ou no pacote mixador, permanecem inalteradas e são adicionadas ao pacote mixador.

Subpacotes dentro do pacote mixado são adicionados ao pacote mixador, caso não existam.

Administração Central

Cetec Capacitações

Se um subpacote com o mesmo nome já existir no pacote mixador, outra mixagem é feita entre os dois subpacotes.

Qualquer importação de pacotes, a partir do pacote mixado, se torna importação do pacote mixador. Elementos que são importados não são mixados (ou seja, não recebem relações de generalização)

Estruturas compostas

Nos modelos UML, um diagrama de estrutura composta mostra a estrutura interna dos classificadores estruturados utilizando peças, portas e conectores. Um classificador estruturado define a implementação de um classificador e pode incluir uma classe, um componente ou um nó de implementação. Você pode utilizar o diagrama de estrutura composta para mostrar os detalhes internos de um classificador e descrever os objetos e funções que trabalham juntos para executar o comportamento do classificador contido.

Um diagrama de estrutura composta é similar a um diagrama de classe, mas ele representa peças individuais em vez de classes inteiras. Antes de definir a estrutura interna de um classificador, você deve mostrar seu compartimento de estrutura ou abrir um diagrama de estrutura composta. Então, você pode modelar as peças que representam as instâncias que o classificador contido possui. Você pode incluir conectores para vincular duas ou mais peças em um relacionamento de associação ou dependência.

Em diagramas da estrutura composta, as portas definem o ponto de interação entre um classificador e seu ambiente ou entre um classificador e suas peças internas. Você pode utilizar uma porta para especificar os serviços que um classificador fornece e requer de seu ambiente.

Você também pode modelar colaborações e usos de colaborações em diagramas da estrutura composta. Uma colaboração descreve as funções e os atributos que definem

Administração Central

Cetec Capacitações

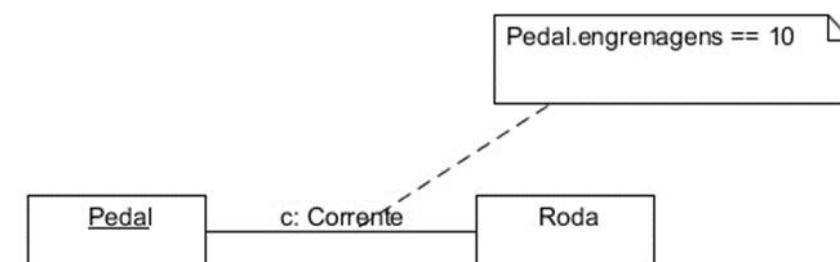
um comportamento específico do classificador. Um uso de colaboração representa um uso específico da colaboração para explicar os relacionamentos entre as propriedades de um classificador. Para identificar as funções das peças no uso de colaboração, você anexa um uso de colaboração a uma colaboração e, em seguida, inclui o uso de colaboração em um diagrama de estrutura composta.

Um diagrama de estrutura composta é exibido no editor de diagrama como uma estrutura que tem o nome do classificador contido. As peças de material composto do classificador e suas conexões de comunicação são exibidas na estrutura do diagrama.

Conectores

Em diagramas UML, um conector é uma linha que representa um relacionamento em um modelo. Ao modelar a estrutura interna de um classificador, você pode utilizar um conector para indicar um link entre duas ou mais instâncias de uma peça ou porta. O conector define o relacionamento entre os objetos ou instâncias que são ligadas a funções no mesmo classificador estruturado e identifica a comunicação entre essas funções. O produto especifica automaticamente o tipo de conector a ser criado.

Figura 46 - Conector



Fonte: O autor.

Administração Central

Cetec Capacitações

Portas

Em diagramas da estrutura composta, uma porta define o ponto de interação entre uma instância do classificador e seu ambiente ou entre o comportamento do classificador e suas peças internas.

Por exemplo, suponha que você queira um subsistema que execute a verificação de pagamentos de cartão de crédito.

A implementação efetiva dessa funcionalidade pode ser dividida em várias classes trabalhando em conjunto.

A organização dessas classes pode ser representada como uma estrutura interna, no âmbito do subsistema, e a funcionalidade geral, ou de verificação do cartão de crédito, pode ser exposta usando uma porta.

Expor a funcionalidade através de uma porta permite que o subsistema possa ser usado por qualquer outro classificador que esteja em conformidade com as especificações da porta.

Execução e Implementação das Portas

Em diagramas da estrutura composta, uma porta define o ponto de interação entre uma instância do classificador e seu ambiente ou entre o comportamento do classificador e suas peças internas.

Devido a todas as interações entre o ambiente externo e as peças internas precisarem passar por uma porta, você pode utilizar uma porta para isolar as peças internas de um objeto de seu ambiente. Conectores vinculam portas a propriedades do classificador e chamam a comunicação entre duas ou mais instâncias. Você pode definir várias portas para um classificador para mostrar interações diferentes dependendo da porta a partir da qual a interação se origina.

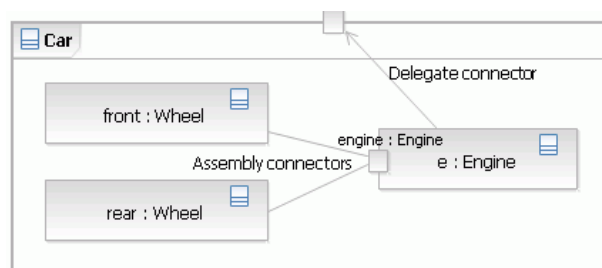
Administração Central

Cetec Capacitações

Uma porta é exibida no quadro de diagramas como um pequeno quadrado com o nome da porta. Você pode incluir portas na moldura de um diagrama de estrutura composta e nas peças internas do diagrama de estrutura composta.

Por exemplo, na figura a seguir, o quadro de diagramas mostra três partes. Duas partes são inseridas com o classificador Wheel e a terceira é inserida pelo classificador Engine. Uma linha reta, chamada de conector de delegação, vincula a parte interna chamada e:Engine a uma porta na extremidade externa do diagrama de estrutura composta. A porta externa é inserida com o classificador Car e é chamada de Port:Car. O conector tem uma b aberta que aponta na direção da porta externa. As duas partes Wheel, chamadas front:Wheel[2] e rear:Wheel[2], são conectadas por linhas retas chamadas de conectores de montagem à porta interna chamada Port:Engine.

Figura 47 – Implementação das Portas



Fonte: IBM [4].

Uma porta pode interagir nas duas direções e você pode incluir interfaces requeridas e fornecidas para especificar os tipos de interações que podem ocorrer entre um classificador e seu ambiente.

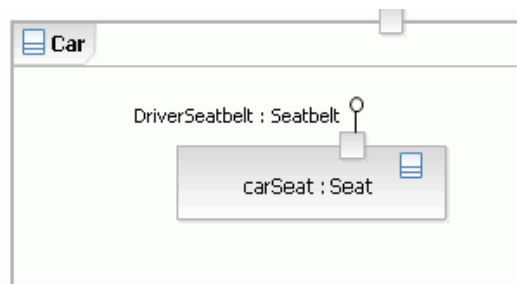
Por exemplo, na seguinte figura, o diagrama exibe um classificador Car que tem uma parte, chamada carSeat:Seat, e duas portas. Uma porta, chamada DriverSeatbelt:Seatbelt, aparece na parte e há um círculo conectado ao topo da porta por uma pequena linha reta. O círculo representa uma interface fornecida, chamada SafetyDevice, que a classe Car oferece para o driver para operar o carro. Na

Administração Central

Cetec Capacitações

extremidade do diagrama de estrutura composta, uma porta, chamada Engine:PowerGenerator, é conectada por uma linha sólida a um meio-círculo. O meio-círculo representa uma interface requerida, chamada Gasoline, que a classe Car precisa para operar o carro.

Figura 48 - Classificador Car



Fonte: IBM [4].

Você pode alterar a posição de uma porta em um diagrama de estrutura composta utilizando a propriedade `isService` da porta. Se a propriedade `isService` for definida como `true`, a porta aparecerá na moldura do diagrama de estrutura composta, indicando que a porta é requerida por seu ambiente. Se você definir a propriedade como `false`, a porta se moverá para dentro do compartimento da estrutura, indicando que a porta é utilizada apenas para implementação interna do classificador e não é requerida por seu ambiente. Quando a propriedade `isService` é configurada como `false`, você pode excluir ou modificar a porta sem afetar o uso do classificador.

Conectores múltiplos

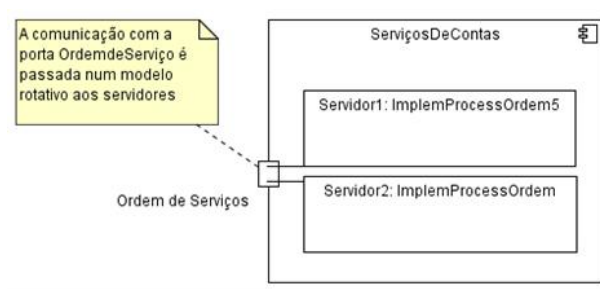
UML 2.0 permite que se tenham vários conectores, ligando uma porta a diferentes elementos internos. No entanto, não especifica o que acontece quando a comunicação é recebida na porta, deixando isso para o moderador. Uma solução possível é passar a comunicação a todos os conectores, com base na prioridade ou na rotatividade, ou

Administração Central

Cetec Capacitações

simplesmente escolher aleatoriamente um conector Independente do que você escolher certifique- se de documentar em seu modelo, provavelmente usando uma nota anexa a porta confira a figura 49:

Figura 49 - Conectores



Fonte: IBM [4].

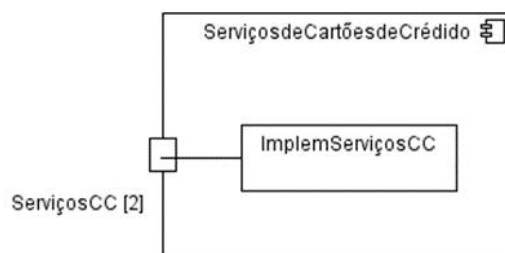
Um classificador pode especificar a multiplicidade de uma porta como qualquer outro elemento, Basta colocar o número desejado de instancias da porta, entre parênteses, após o nome da porta e o tipo (se houver). Quando o classificador for instanciado, as portas associadas serão instanciadas também.

Isto se chama pontos de interação e pode ser identificado exclusivamente pelo classificador. Por exemplo, se o classificador tem duas portas, ambas com interfaces fornecidas exclusivamente pelo classificador. Por exemplo, se o classificador tem duas portas, ambas com interfaces fornecidas – uma oferece acesso anônimo a dados e a outra oferece acesso autenticado aos dados – o classificador pode distinguir qual porta foi usada pelo sistema externo. A figura 50 mostra um sistema de verificação de cartão de crédito que oferece duas instancias da porta de verificação de cartão de credito.

Administração Central

Cetec Capacitações

Figura 50 - Sistema de Verificação Cartão



Fonte: O autor.

Administração Central

Cetec Capacitações

Cap. 6 Diagramas de Componentes

Na modelagem UML, os componentes são elementos de modelo que representam partes independentes e intercambiáveis de um sistema. Eles estão em conformidade e realizam uma ou mais interfaces fornecidas e requeridas, que determinam o comportamento dos componentes.

Os componentes tornam um sistema mais flexível, escalável e reutilizável.

Para que um componente seja substituível, ele deve atender aos seguintes critérios:

- A estrutura interna do componente deve ser oculta. Não pode existir nenhuma dependência entre o conteúdo do componente e de outros objetos.
- Os componentes devem fornecer interfaces para que os objetos externos possam interagir com eles.
- A estrutura interna do componente deve ser independente. Os objetos internos não devem conhecer os objetos externos.
- Os componentes devem especificar suas interfaces requeridas para que eles tenham acesso a objetos externos.

Nos modelos que descrevem sistemas executáveis, os componentes representam os componentes utilizados durante a execução do sistema. Exemplos incluem objetos COM+, JavaBeans e serviços da Web.

Normalmente, um componente recebe o nome da parte do sistema que ele representa.

Como a figura a seguir ilustra, um componente é exibido no editor de diagrama como um retângulo que contém o nome do componente. Ela também contém um estereótipo «component» ou um ícone de componente, que é uma caixa com dois pequenos retângulos que sobressaem de seu lado.

Administração Central

Cetec Capacitações

Figura 51 - Componente



Fonte: O autor

Os compartimentos são usados para exibir informações sobre os atributos, operações, interfaces fornecidas, interfaces necessárias, realizações e estrutura interna do componente.

Instancias de Componentes

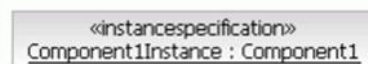
Na modelagem UML, as instâncias de componentes são elementos de modelo que representam entidades reais em um sistema.

Normalmente, as instâncias de componentes são utilizadas em diagramas de implementação para representar unidades de execução que existem em tempo de execução; entretanto, também é possível utilizá-las em diagramas de componentes.

O nome de uma instância de componente consiste em uma concatenação sublinhada do nome da instância, dois pontos (:) e o nome do componente; por exemplo, Shopper1:Cart.

Como ilustrado na figura 52, uma instância de componente é exibida no editor de diagrama como um retângulo que contém o nome e o estereótipo.

Figura 52 - Instância de Componente



Fonte: O autor.

Administração Central

Cetec Capacitações

Exemplo

Você está desenvolvendo um aplicativo de e-commerce que distribui componentes entre o navegador da Web do cliente, um servidor da Web público e um servidor de dados particular. Para especificar em qual dispositivo um componente específico (por exemplo, o componente Cart) é executado, é possível criar um diagrama de implementação com três instâncias de nó: um para o computador que executa o navegador da Web, outro para o servidor da Web e um terceiro para o servidor de dados. Em seguida, é possível criar um relacionamento de implementação entre a instância do componente Cart e a instância do nó do servidor da Web para indicar que a instância do componente é executada no servidor da Web.

Artefatos

Em modelos UML, *artefatos* são elementos de modelo que representam as entidades físicas em um sistema de software. Os artefatos representam unidades físicas de execução, como por exemplo arquivos executáveis, bibliotecas, componentes de software, documentos e bancos de dados.

Os artefatos são geralmente utilizados em diagramas de implementação, mas você pode utilizá-los também em diagramas de componentes para mostrar os elementos de modelo, como componentes ou classes, que são manifestados no artefato. Os elementos de modelo podem ser manifestados em diversos artefatos diferentes.

Os artefatos são implementados em nós e especificam as unidades físicas de informações que a execução e operação de um sistema utilizam ou produzem. Os artefatos podem ser suportados para implementação em diversos tipos de nós.

Nos diagramas, as divisões exibem informações sobre os atributos e operações do artefato.

Administração Central

Cetec Capacitações

Um artefato possui um nome exclusivo que descreve o arquivo ou o componente de software que ele representa.

Como a figura 53 ilustra, um artefato é exibido como um retângulo que contém o nome do artefato. O retângulo também contém o estereótipo «artifact» e o ícone do artefato.

Figura 53 - Artefato

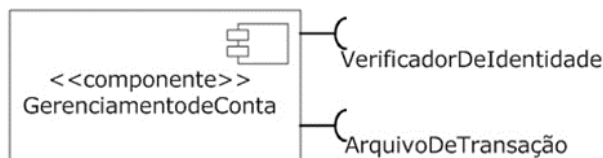


Fonte: IBM [4].

Conectores de montagem

Quando se modela a visão caixa-preta de um componente, as interfaces fornecidas e requeridas são representadas usando-se conectores de montagem. Os conectores são ilustrados por ícones bola e soquete. Para mostrar uma interface requerida, use o ícone soquete e escreva o nome da interface junto ao símbolo do conector, como mostrado no exemplo:

Figura 54 - Conectores de montagem



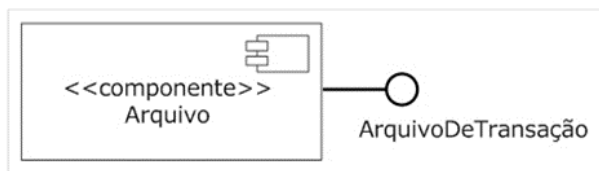
Fonte: O autor.

Para mostrar uma interface fornecida, utilize uma bola de um conector de montagem, novamente com o nome da interface junto ao símbolo como mostra a figura 55:

Administração Central

Cetec Capacitações

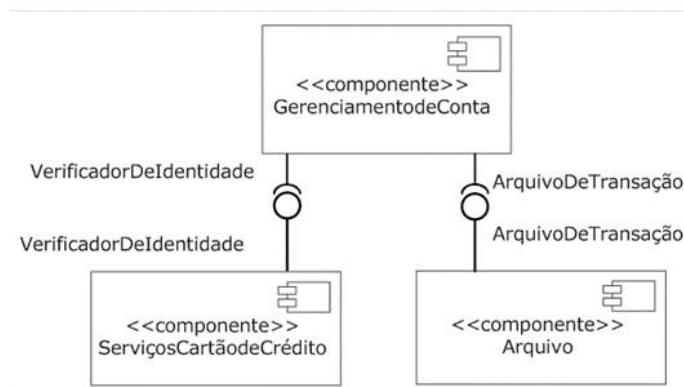
Figura 55 - Interface



Fonte: O autor.

Para conectar componentes, basta ligar o acoplamento de interfaces fornecidas e requeridas. As dependências de componente, usando conectores de montagem, fornecem mais detalhes sobre as relações entre componentes do que as relações de dependência simples. Veja o exemplo conforme figura 56:

Figura 56 - Conectando Componentes



Fonte: IBM [4].

Administração Central

Cetec Capacitações

Compartilhamento de Componentes

UML também fornece uma visão caixa-preta de componentes usando compartilhamentos. É possível adicionar um compartilhamento para mostrar interfaces fornecidas e requeridas. Rotule as interfaces com o estereótipo <<interfaces fornecidas>> e as interfaces requeridas com o estereótipo <<interfaces requeridas>>, como mostra o exemplo a seguir:

Figura 57 - Compartilhamento de Componentes



Fonte: IBM [4].

Caixa Branca

A fim de fornecer detalhes sobre implementação de um componente, UML define uma visão caixa branca, que mostra exatamente como determinado componente executa as interfaces que fornece. Isso geralmente é feito usando classes e ilustrando com um diagrama de classe. Entretanto, um componente pode delegar parte, ou a totalidade do seu comportamento, a outros componentes.

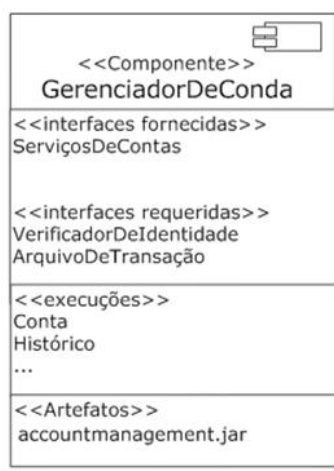
A mais simples visão caixa-branca de um componente é obtido acrescentando um compartimento ao componente e listando os classificadores que o executam. O compartilhamento deve ser rotulado com o estereótipo <<execuções>>. Embora isso

Administração Central

Cetec Capacitações

forneça mais detalhes do que uma visão caixa-preta é de uso limitado para os desenvolvedores de componentes.

Figura 58 - Caixa Branca



Fonte: O autor.

Dependência de classificador

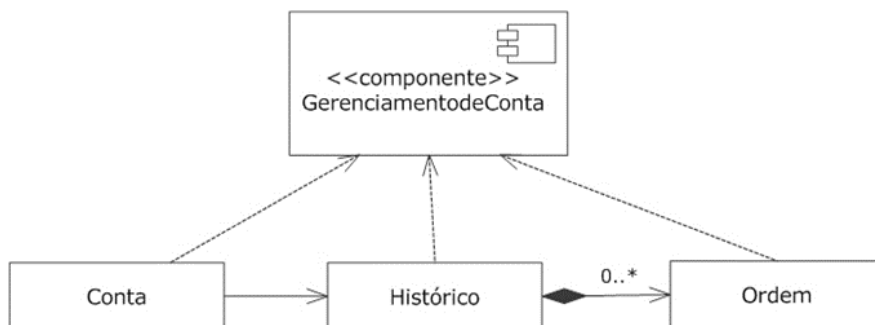
Para mostrar a parte interna de um componente, mostre cada classificador que o excuta com uma dependência do próprio componente.

Observe que a relação entre os classificadores e os componentes de dependência (linha tracejada, seta aberta) e não uma relação de execução. Essa notação é útil para identificar que constituem um componente, mas o foco do esquema ainda é o componente como um todo. A imagem abaixo representa a visão caixa branca de um componente e seus classificadores constitutivos:

Administração Central

Cetec Capacitações

Figura 59 - Dependência de Classificador



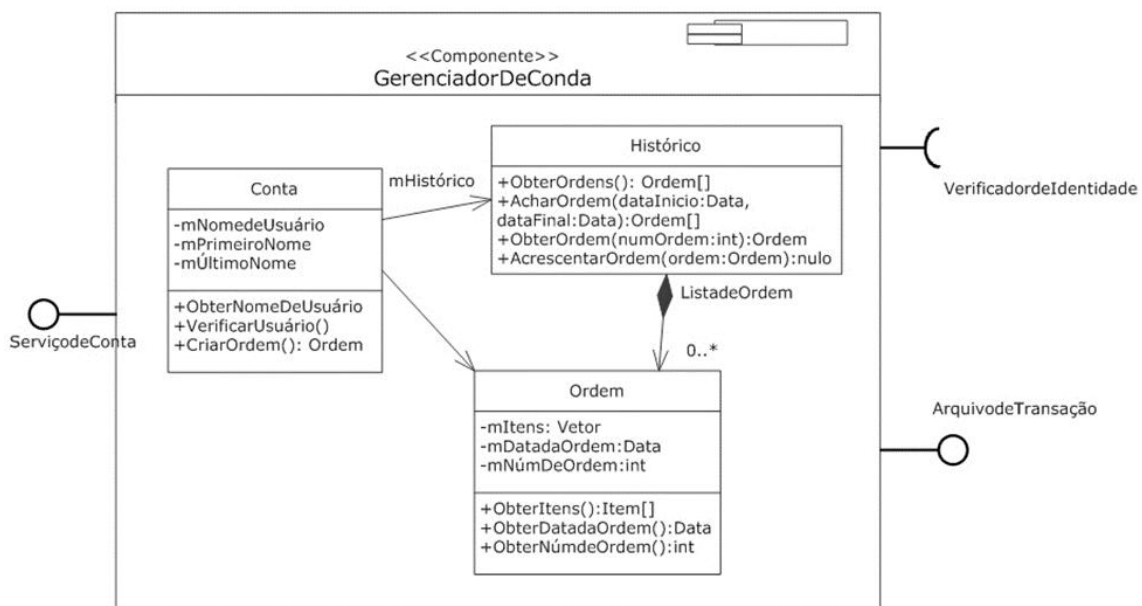
Fonte: IBM [4].

Para mudar o foco para estrutura dos classificadores, basta mostra-los dentro retângulo do componente. Isso tem o efeito de enfatizar as relações dos classificadores que compõem o componente, além de incentivar o encapsulamento de componentes. Veja na figura 60.

Administração Central

Cetec Capacitações

Figura 60 - Relações dos Classificadores



Fonte: IBM [4].

Se a parte interna de um componente é complexa, é comum usar um diagrama de classes separado para modelar os detalhes. O novo diagrama de classe pode ser ligado a seus componentes usando nota.

Portas e conectores

A UML introduz o conceito de portas para que possa identificar explicitamente a funcionalidade que é exposta ao mundo exterior.

Uma porta agrupa interfaces requeridas e fornecidas relacionadas e usa conectores para mapeá-las para um classificador que executa a funcionalidade. Se uma porta tem tanto interfaces fornecidas como requeridas, é chamada de portas bidirecional. Aporta

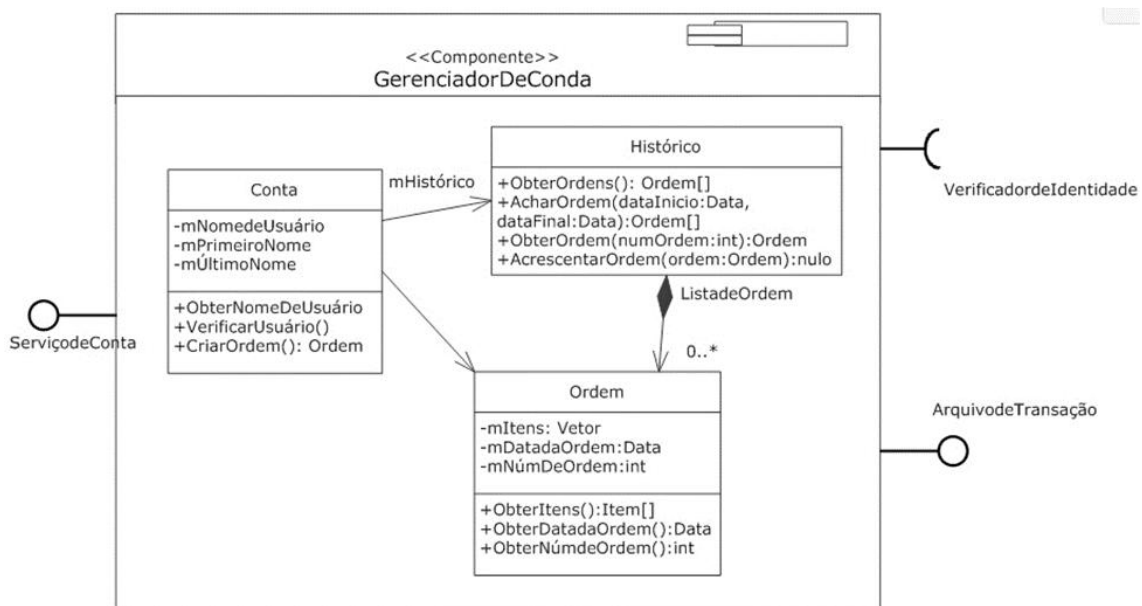
Administração Central

Cetec Capacitações

é mostrada por um pequeno retângulo em um dos lados de um classificador. Um conector de montagem (bola e soquete) indica as interfaces fornecidas e requeridas.

A fim de demonstrar a execução da funcionalidade, um conector mapeia a porta para um classificador interno. UM conector é mostrado por uma linha continua com uma seta cheia, da porta para o classificador. Um conector indica que todas as mensagens que chegam a porta (geralmente sob a forma de chamadas de métodos) são encaminhadas para o classificador específico. Também é possível usar conectores, do classificador para a porta, mostrar as mensagens sendo passadas através de interfaces fornecida. A figura 61 mostra a visão caixa-branca de um componente com três portas:

Figura 61 - Visão caixa-branca



Fonte: IBM [4].

Administração Central

Cetec Capacitações

Estereótipos de Componentes

UML define vários estereótipos que se aplicam especificamente aos elementos.

Entidade

É um componente que representa um conceito de negócio. O componente entidade, basicamente transmite informações dentro e fora de interfaces e muitas vezes é usado como um todo. As entidades normalmente não têm qualquer funcionalidade ou aptidão para serviços associados a elas, e geralmente existem apenas para armazenar e recuperar dados.

Processo

É um componente que pode preencher requisitos funcionais (em oposição ao componente entidade). O componente processo é baseado em transação e normalmente tem algum tipo de estado associado a ele (ao contrário de componentes de serviço sem estado).

Execução

É um componente que não tem especificação própria. Ao contrário, é a execução de um componente especificação, como mostrado na figura 62:

Administração Central

Cetec Capacitações

Figura 62 - Execução



Fonte: IBM [4].

Serviço

É um componente em estado, que pode preencher requisitos funcionais. Os componentes de serviço são raramente usados, pois não possuem qualquer estado.

Especificação

É um componente que tem interfaces fornecidas e requeridas, mas não implementação. A componente especificação, mostrado na imagem acima, deve formar par com um componente de execução ou componente implementar.

Subsistema

Este componente faz parte de um sistema maior. A UML não fornece a definição real de um subsistema, no entanto, geralmente um subsistema significa um conjunto autocontido de funcionalidade, maior do que um simples componente.

Administração Central

Cetec Capacitações

Cap. 7 Diagramas de Casos de Uso

Na UML, os diagramas de casos de uso modelam o comportamento de um sistema e ajudam a capturar os requisitos do sistema.

Os diagramas de casos de uso descrevem funções de alto nível e escopo de um sistema. Esses diagramas também identificam as interações entre o sistema e seus agentes. Os casos de uso e os agentes nos diagramas de casos de uso descrevem o que o sistema faz e como os agentes o usam, mas não como o sistema opera internamente.

Os diagramas de casos de uso ilustram e definem o contexto e os requisitos de um sistema inteiro ou das partes importantes dele. É possível modelar um sistema complexo com um único diagrama de caso de uso ou ainda criar muitos diagramas de casos de uso para modelar os componentes do sistema. Normalmente, os diagramas de casos de uso são desenvolvidos nas fases iniciais de um projeto e são consultados em todo o processo de desenvolvimento.

Os diagramas de casos de uso são úteis nas seguintes situações:

- Antes de iniciar um projeto, é possível criar diagramas de casos de uso para modelar um negócio de forma que todos os participantes no projeto compartilhem um entendimento dos trabalhadores, clientes e atividades do negócio.
- Ao reunir os requisitos, é possível criar diagramas de casos de uso para capturar os requisitos do sistema e apresentar a terceiros o que o sistema deve fazer.
- Durante as fases de análise e design, é possível criar casos de uso e agentes nos diagramas de casos de uso para identificar as classes que o sistema requer.
- Durante a fase de teste, é possível utilizar os diagramas de casos de uso para identificar testes para o sistema.

Administração Central

Cetec Capacitações

Os seguintes tópicos descrevem elementos de modelos em diagramas de casos de uso:

- **Casos de Uso**
Um caso de uso descreve uma função que um sistema desempenha para alcançar a meta do usuário. Um caso de uso deve produzir um resultado observável que seja valioso para o usuário do sistema.
- **Agentes**
Um agente representa uma função de um usuário que interage com o sistema que está sendo modelado. O usuário pode ser um humano, uma organização, uma máquina ou outro sistema externo.
- **Subsistemas**
Nos modelos UML, os subsistemas são um tipo de componente estereotipado que representa unidades comportamentais independentes em um sistema. Os subsistemas são utilizados em diagramas de classe, componentes e de caso de uso para representar componentes de larga escala no sistema que está sendo modelado.
- **Relacionamentos em Diagramas de Caso de Uso**
Na UML, um relacionamento é uma conexão entre elementos de modelo. Um relacionamento UML é um tipo de elementos de modelo que inclui semântica em um modelo, definindo a estrutura e o comportamento entre os elementos de modelo.

Um caso de uso descreve uma função que um sistema desempenha para alcançar a meta do usuário. Um caso de uso deve produzir um resultado observável que seja valioso para o usuário do sistema.

Os casos de uso contêm informações detalhadas sobre o sistema, os usuários do sistema, os relacionamentos entre o sistema e os usuários e o comportamento requerido do sistema. Os casos de uso não descrevem os detalhes de como o sistema é executado.

Administração Central

Cetec Capacitações

Cada caso de uso descreve uma meta específica para o usuário e como o usuário interage com o sistema para atingir tal meta. O caso de uso descreve todas as formas possíveis em que o sistema pode alcançar, ou falhar em alcançar, a meta do usuário.

É possível utilizar casos de uso para os seguintes objetivos:

- Determinar os requisitos do sistema
- Descrever o que o sistema deve fazer
- Fornece uma base para teste, para assegurar que o sistema funcione conforme pretendido

Nos modelos que descrevem negócios, os casos de uso representam os processos e atividades do negócio. Nos modelos que descrevem sistemas de software, os casos de uso representam os recursos do software.

Caso de Uso

Cada caso de uso deve possuir um nome exclusivo que descreva a ação que o sistema desempenha. Normalmente, os nomes de caso de uso são frases curtas que iniciam com um verbo, como Efetuar Pedido On-line. Como a figura 63 ilustra, um caso de uso é exibido como um oval que contém o nome do caso de uso.

Figura 63 - Caso de Uso



Fonte: O autor.

Administração Central

Cetec Capacitações

É possível incluir os seguintes recursos em casos de uso:

- Atributos que identificam as propriedades dos objetos em um caso de uso
- Operações que descrevem o comportamento dos objetos em um caso de uso e como eles afetam o sistema
- Documentação que detalha o objetivo e o fluxo de eventos em um caso de uso

Agentes

Um agente representa uma função de um usuário que interage com o sistema que está sendo modelado. O usuário pode ser um humano, uma organização, uma máquina ou outro sistema externo.

É possível representar vários usuários com um único agente e um único usuário pode ter a função de vários agentes. Os agentes são externos ao sistema. Eles podem iniciar o comportamento descrito no caso de uso ou sofrerem ação pelo caso de uso. Os agentes também podem trocar dados com o sistema.

Em modelos que descrevem negócios, os agentes representam os tipos de indivíduos e máquinas que interagem com um negócio. E modelos que descrevem aplicativos de software, os agentes representam os tipos de indivíduos, sistemas externos ou máquinas que interagem com o sistema.

Normalmente, os agentes são utilizados em diagramas de casos de uso, mas também é possível utilizá-los em diagramas de classe e de sequência. Como a figura 64 ilustra, um agente é exibido como um desenho em linha de uma pessoa.

Figura 64 - Agente



Fonte: IBM [4].

Administração Central

Cetec Capacitações

Cada agente possui um nome exclusivo que descreve a função do usuário que interage com o sistema.

Você pode incluir documentação que defina a função do agente e como ele interage com o sistema.

Pacotes

Os pacotes agrupam elementos de modelos relacionados de todos os tipos, incluindo outros pacotes.

É possível agrupar elementos de modelo em pacotes com os seguintes motivos:

- Organizar os elementos de modelo para que o modelo seja mais fácil de entender e de navegar no Explorador de Projetos.
- Modelar a arquitetura do sistema utilizando pacotes para representar as diversas camadas ou subsistemas

Os pacotes também representam espaços de nomes, o que significa que os elementos de modelo em um pacote devem possuir nomes exclusivos. Por exemplo, se você validar um modelo que possua um agente denominado Customer e uma classe denominada Customer no mesmo pacote, receberá um aviso de que aparecem nomes conflitantes no mesmo pacote. Para impedir o aviso, é possível colocar o agente Customer em outro pacote.

É possível utilizar pacotes em diversos diagramas, incluindo diagramas de classe, de componentes e de caso de uso. Cada pacote possui um nome exclusivo que descreve seu conteúdo.

Como a figura 64 ilustra, um pacote é exibido como um retângulo com uma guia no canto superior esquerdo. O retângulo contém o nome do pacote e o ícone do pacote.

Administração Central

Cetec Capacitações

Figura 65 - Pacote



Fonte: IBM [4].

Subsistemas

Nos modelos UML, os subsistemas são um tipo de componente estereotipado que representa unidades comportamentais independentes em um sistema. Os subsistemas são utilizados em diagramas de classe, componentes e de caso de uso para representar componentes de larga escala no sistema que está sendo modelado.

É possível modelar um sistema inteiro como uma hierarquia de subsistemas. Também é possível definir o comportamento que cada subsistema representa especificando interfaces para os subsistemas e as operações que suportam as interfaces.

Nos diagramas, as divisões exibem informações sobre os atributos, operações, interfaces fornecidas, interfaces requeridas, realizações e estrutura interna do subsistema.

Normalmente, um subsistema possui um nome que descreve seu conteúdo e função no sistema.

Limites do Sistema

Por definição, os casos de uso capturam funcionalidade de terminado sujeito. Qualquer coisa que seja executada pelo sujeito é considerada fora dos limites do sistema e deve ser modelado como um ator. Essa técnica é bastante útil para

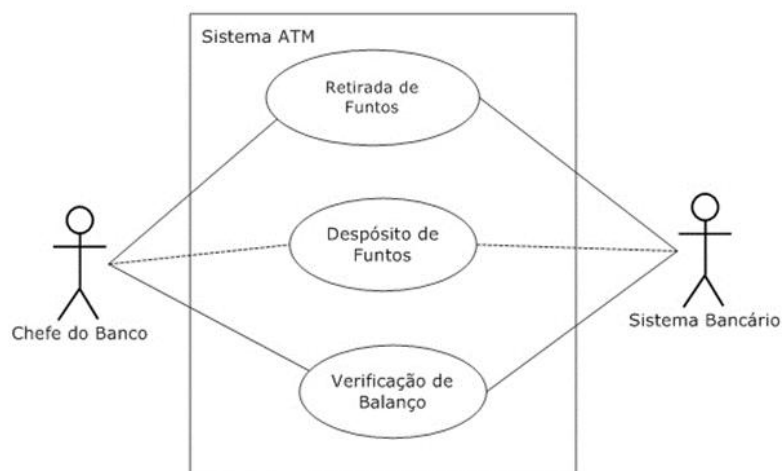
Administração Central

Cetec Capacitações

determinar o alcance e a atribuição de responsabilidade quando se projeta um sistema, subsistema ou componente. Por exemplo, ao modelar um sistema de ATM (caixa de banco), as discussões de projeto se desviam para os detalhes do terminal do sistema bancário, um modelo de caso de uso com os limites do sistema claramente definidos, que identifiquem o sistema bancário como um ator e assim expõe o escopo do problema.

Os limites do sistema são representados em sentido genérico, usando um simples retângulo com o nome do sistema no topo:

Figura 66 - Exemplo de Caso de Uso



Fonte: O autor.

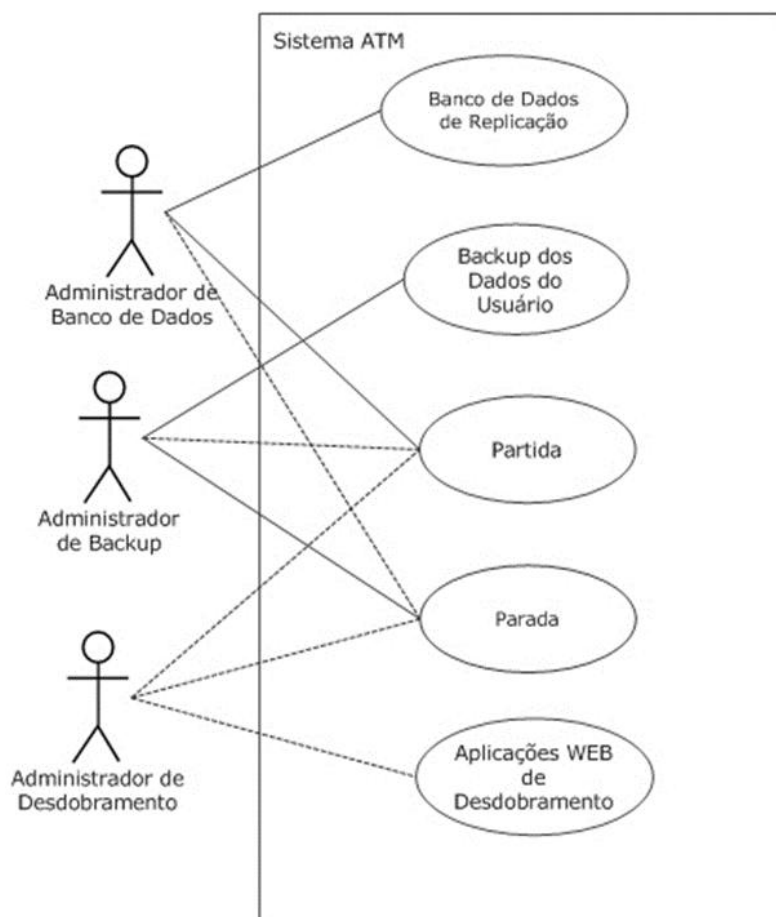
Os atores não precisam ter o mapeamento um-para-um para as entidades físicas. Na verdade, eles não precisam ser entidades físicas.

Administração Central

Cetec Capacitações

UML permite que os atores representem papéis de usuários potenciais de um sistema. Por exemplo, o administrador pode ser o único usuário físico do sistema e ainda assim, ter muitas especialidades. Pode ser útil ver o sistema a partir da perspectiva do administrador de banco de dados, do administrador de backup, do administrador de implantação e assim por diante.

Figura 67 - Caso de Uso - Sistema ATM



Fonte: O autor.

Administração Central

Cetec Capacitações

Modelagem Avançada de Casos de Uso

A UML fornece mecanismo para reutilizar e acrescentar casos de uso e atores. É possível expandir de um ou substituir os casos de uso, usando generalização. Para expandir elementos comuns de caso, incluídos ou adicione a base de casos de uso usando a extensão.

Generalização de ator e Caso de Uso

A generalização do ator é normalmente usada para extrair os requisitos comuns de vários atores diferentes para simplificar a modelagem.

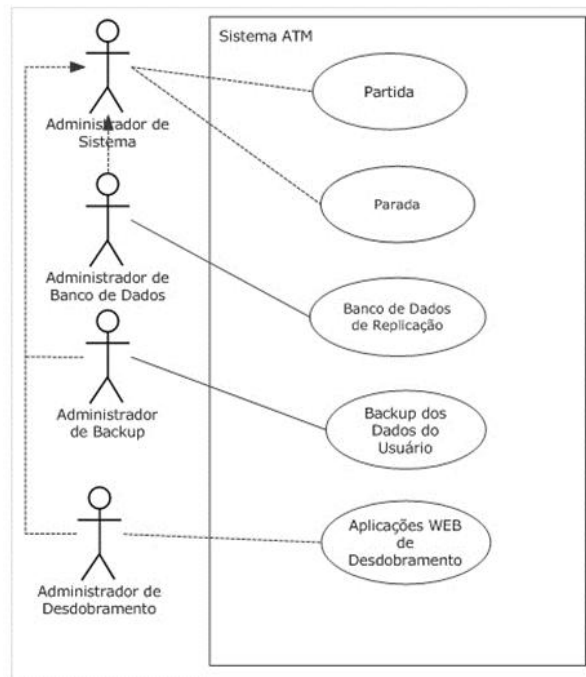
Um ator Administrador de Sistema genérico pode ser destacado para extrair as funcionalidades comuns e em seguida, especializa-lo para as necessidades específicas de cada ator.

A generalização do ator é representada como a de qualquer outro classificador é traçada uma linha continua com uma seta fechada, que aponta do ator especializado para o ator base. Confira a figura 68:

Administração Central

Cetec Capacitações

Figura 68 - Generalização



Fonte: IBM [4].

Administração Central

Cetec Capacitações

Cap. 8 Máquina de Estado

Na modelagem UML, uma máquina de estado é uma especificação do comportamento dinâmico de objetos de classe individuais, casos de uso e sistemas inteiros. Com a exceção das operações, ao criar uma máquina de estado, o objeto anexado à máquina de estado se torna seu proprietário. Ao criar uma máquina de estado para uma operação, está se torna o proprietário da máquina de estado. Um diagrama de máquina de estado em branco é aberto quando você cria uma máquina de estado. Um diagrama de máquina de estado é uma representação gráfica da sequência de estados de um objeto, dos eventos que causam uma transição de um estado para outro e as ações que resultam de uma alteração no estado. É possível incluir diagramas em uma máquina de estado para descrever diferentes aspectos comportamentais de um objeto.

É possível criar máquinas de estado para descrever classes e sistemas que possuem comportamento significativo. Nem todos os objetos requerem máquinas de estado. Se o comportamento de um objeto for básico, caso ele simplesmente armazene ou recupere dados, o comportamento do objeto pode não ser importante para você e sua máquina de estado pode ser de pouco interesse. As máquinas de estado também podem conter estados aninhados que representam níveis de estado hierárquico diferentes. É possível utilizar estados aninhados para examinar alterações complexas de estado em objetos.

É possível incluir diagramas em uma máquina de estado para descrever diferentes perspectivas do comportamento de um objeto. Cada diagrama é aberto como uma janela separada, mas os mesmos elementos de modelo são exibidos em todos os diagramas. Os diagramas em uma máquina de estado são sincronizados por padrão. As alterações feitas em uma região na visualização Explorador de Projetos são refletivas em outros diagramas da mesma máquina de estado e alterações feitas em uma região em um diagrama são refletidas na visualização Explorador de Projetos. É possível alterar as configurações de edição de uma região para que as alterações feitas em uma

Administração Central

Cetec Capacitações

região na visualização Explorador de Projetos não sejam refletidas nos diagramas de máquina de estado correspondentes e para que uma região possa ser editada independentemente de outros diagramas na mesma máquina de estado, alterando o valor das propriedades canônicas de uma região para false.

As máquinas de estado são auxílios de modelagem úteis para o desenvolvimento de sistemas de tempo real ou dirigidos por eventos porque elas mostram o comportamento dinâmico. É possível desenvolver máquinas de estado durante todas as fases de um projeto de software e para modelagem de negócios. É possível utilizar máquinas de estado nas seguintes situações:

- Durante a modelagem do negócio, é possível criar máquinas de estado para modelar um cenário de caso de uso.
- Durante a análise e o design, é possível modelar objetos dirigidos por eventos que reagem a eventos externos ao contexto de um objeto.
- Durante a análise e o design, é possível utilizar diversos diagramas de máquina de estado para mostrar aspectos diferentes da mesma máquina de estado e seu comportamento.

É possível criar uma máquina de estado para os seguintes objetos:

- Classes
- Colaborações
- Componentes
- Nós
- Operações
- Casos de Uso

Administração Central

Cetec Capacitações

Comportamento Dirigido por Eventos

As máquinas de estado podem ser utilizadas para modelar comportamento dirigido por eventos. Eventos como tempo, sinais ou operações podem fazer com que o estado de um objeto seja alterado. Um evento não possui duração e pode preceder ou seguir outro evento. Os estados que modelam o comportamento dirigido por eventos continuam os mesmos até a chegada de um evento. Após o estado responder a um evento, o processo reverte a um estado estável de que está pronto para receber o próximo evento.

Criando transições entre Estados

Em modelagem UML, é possível incluir transições em um diagrama de máquina de estado para mostrar como um objeto altera o estado. Um acionador, uma condição de proteção e um efeito são as três partes opcionais de uma transição. Inclua um acionador em uma transição para mostrar que um evento deve ocorrer para que uma transição seja iniciada. Inclua uma condição de proteção em uma transição para mostrar que uma condição Booleana deve ser verdadeira para que ocorra uma transição. Inclua um efeito em uma transição para mostrar que um objeto executa uma atividade específica quando uma condição de proteção é atendida.

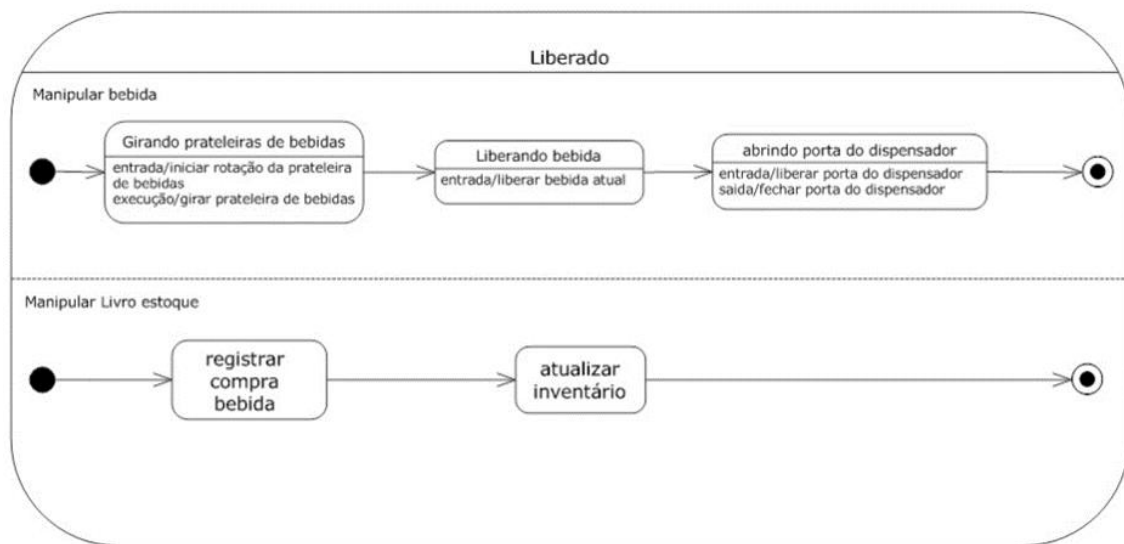
Regiões

Uma região é mostrada por uma linha pontilhada, dividindo o compartimento de decomposição. Cada região pode ser nomeada escrevendo seu nome dentro da área da região, como mostra o exemplo na figura 69:

Administração Central

Cetec Capacitações

Figura 69 - Regiões



Fonte: IBM [4].

Cada região tem o próprio pseudoestado inicial e o estado final. A transição para o estado composto é uma transição para o pseudocódigo inicial de cada região. Cada região de um estado composto é executada em paralelo, sendo perfeitamente aceitável que uma região termine antes de outra. A transição para estado final de uma região indica que a atividade para essa região foi completada. Depois de ter completado todas as regiões, o estado composto dispara um evento de conclusão e uma transição de conclusão (se houver) é disparada.

Estado de Submáquina

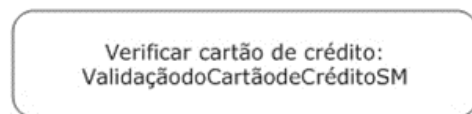
Usados de submáquinas são semanticamente equivalentes aos estados compostos a medida que são feitas de subestado internos e transições. UML define o estado de submáquina como forma de encapsular estados e transições para que possam ser reutilizados. Um estado submáquina significa simplesmente que outra máquina é de estado, um estado de submáquina é contido pelo Estado

Administração Central

Cetec Capacitações

O estado submáquina também é mostrado no retângulo arredondado como qualquer outro estado. Exceto o nome do estado, que é mostrado seguido por dois pontos (:), seguido do nome da submáquina referenciada. Verifique o exemplo a seguir:

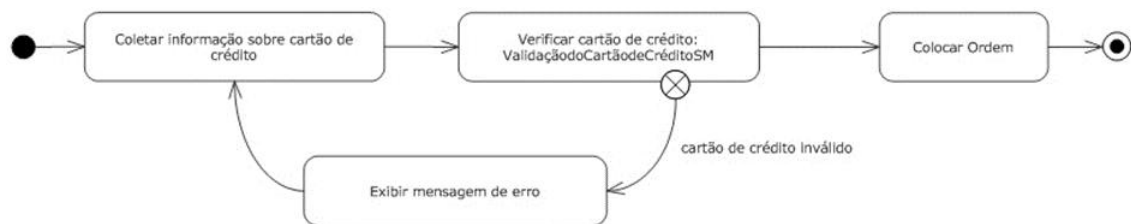
Figura 70 - Submáquina



Fonte: O autor

Normalmente, quando se mostra um estado de submáquina, os pontos de entrada e saída da referida submáquina são mostrados como na figura 71:

Figura 71 - Estado de submáquina



Fonte: IBM [4].

Se o estado de submáquina é introduzido através do pseudoestado padrão inicial, ou retirado por causa da conclusão da submáquina, os pontos de entrada/saída não precisam ser mostrados de forma explícita.

Para maior clareza o mesmo ícone composto deve ser usado para mostrar que a submáquina referenciada é definida em outras partes do modelo.

Administração Central

Cetec Capacitações

Criando Transições entre Estados

Em modelagem UML, é possível incluir transições em um diagrama de máquina de estado para mostrar como um objeto altera o estado. Um acionador, uma condição de proteção e um efeito são as três partes opcionais de uma transição. Inclua um acionador em uma transição para mostrar que um evento deve ocorrer para que uma transição seja iniciada. Inclua uma condição de proteção em uma transição para mostrar que uma condição Booleana deve ser verdadeira para que ocorra uma transição. Inclua um efeito em uma transição para mostrar que um objeto executa uma atividade específica quando uma condição de proteção é atendida.

Estados, Regiões e Transições

Na modelagem UML, os estados representam o comportamento de alteração de um objeto. Uma alteração de estado é descrita utilizando uma transição para mostrar um caminho entre dois estados.

Estados

Um estado pode conter outros estados, normalmente chamados de estados aninhados ou subestados. Se estiver modelando máquinas de estado complexas, utilize estados aninhados para separar o comportamento detalhado em múltiplos níveis. Os estados também podem conter ações que identifiquem as tarefas que podem ocorrer quando um objeto possui um estado específico.

Tipos de Estado e Regiões



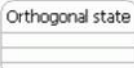


Cada estado é dividido em divisões. A divisão superior exibe o nome do estado. Abaixo da divisão do nome fica a divisão de ação. A divisão de ação exibe as atividades de execução, entrada ou saída que o estado pode conter. Cada divisão abaixo da divisão

Administração Central

Cetec Capacitações

de atividades representa uma região. Máquinas de estado, estados compostos e estados ortogonais contêm regiões. Uma região pode conter estados, pseudo-estados e transições. Utilize regiões para definir estados e transições aninhados.

Tabela 4 - Tipos de Estados

Tipo de estado	Descrição	Elemento do diagrama
Simples	Um estado sem regiões	
Composto	Um estado com uma região	
Ortogonal	Um estado com duas ou mais regiões	
Final	Um estado que é colocado na região de um estado composto para indicar que a atividade na região está concluída	
Estado da submáquina	Um estado que faz referência a outra máquina de estado	

Fonte: IBM [4].

Transições

Uma transição mostra um caminho entre estados que indica que está ocorrendo uma alteração de estado. Um acionador, uma condição de proteção e um efeito são as três partes de uma transição, todas as quais são opcionais.

Um acionador é um evento que deve ocorrer para que uma transição seja iniciada. Uma condição de proteção é uma condição Booleana que deve ser verdadeira para que uma transição ocorra. Um efeito é uma ação ou atividade que o objeto executa quando uma condição de proteção é satisfeita.

Administração Central

Cetec Capacitações

Tabela 5 - Tipos de Estados

Tipo de evento	Descrição
Chamada	Um objeto recebe um pedido para chamar uma operação. A chamada da operação aciona uma transição.
Alteração	Uma condição booleana é específica para acionar uma transição quando a condição for verdadeira.
Sinal	Uma mensagem especificada que aciona uma transição quando é recebida por um objeto.
Tempo	Um período de tempo especificado que deve decorrer ou um tempo absoluto que aciona uma transição.

Fonte: IBM [4].

Administração Central

Cetec Capacitações

Cap. 9 Diagrama de Atividade

Um diagrama de atividade mostra um processo de negócios ou fornece uma visualização de comportamento de um sistema, descrevendo a sequência de ações em um processo. Os diagramas de atividades são parecidos aos fluxogramas, pois eles mostram o fluxo entre as ações em uma atividade.

O Diagrama de Atividades é uma ferramenta poderosa que facilita a compreensão e análise do sistema do negócio, podemos usá-la para analisar um caso de uso, para compreender um determinado workflow, descrever um determinado algoritmo complicado e também lidar com aplicações de processamento paralelo

Podemos usar um diagrama de atividade para descrever os seguintes processos:

- Um processo de negócios ou um fluxo de trabalho entre usuários e seu sistema. Para obter mais informações.
- As etapas executadas em um caso de uso.
- Um protocolo de software, ou seja, as sequências permitidas de interações entre os componentes.
- Um algoritmo de software.

Diagramas de atividade de leitura

As ações e outros elementos que aparecem em um diagrama de atividade formam uma atividade, podemos ver essa atividade no Gerenciador de modelos UML, quando adicionamos o primeiro elemento no diagrama ele é criado.

Para ler um diagrama, imagine que um token ou thread de controle, passa os conectores de uma ação para a próxima.

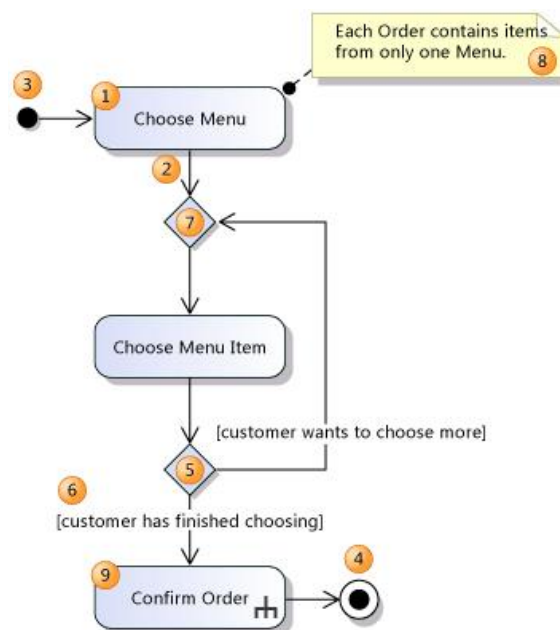
Fluxos de controle simples

Administração Central

Cetec Capacitações

Você pode mostrar uma sequência de ações com loops e ramificações.

Figura 72 - Sequência de ações



Fonte: IBM [4].

Tabela 6 - Elementos e sua Descrição

Forma	Elemento	Descrição e propriedades principais
1	Ação	<p>Uma etapa na atividade, na qual os usuários ou software executar alguma tarefa.</p> <p>A ação pode iniciar quando um token tiver atingido todos os seus fluxos de entrada. Ao terminar, os tokens são enviados em todos os fluxos de saída.</p>

Administração Central

Cetec Capacitações

		<ul style="list-style-type: none"> • Corpo -Especifica a ação em detalhes. • Idioma -o idioma da expressão no corpo. • Pós-condições local -restrições que devem ser atendidas quando termina a execução. O objetivo obtido pela ação. • Pré-condições local -restrições que devem ser atendidas antes de inicia a execução.
2	Fluxo de Controle	<p>Um conector que mostra o fluxo de controle entre as ações. Para interpretar o diagrama, imagine que um token flui de uma ação para o próximo.</p> <p>Para criar um fluxo de controle, use a ferramenta Conector.</p>
3	Nó Inicial	Indica a primeira ação ou ações na atividade. Quando a atividade é iniciada, um token flui a partir do nó inicial.
4	Nó Final da Atividade	Um final para a atividade. Quando um token chega, a atividade será finalizada.
5	Nó de Decisão	Uma ramificação condicional em um fluxo. Tem uma entrada e duas ou mais saídas. Um token de entrada surge em apenas uma das saídas.
6	Proteção	Uma condição que especifica se um token pode fluir ao longo de um conector. Usados com mais frequência

Administração Central

Cetec Capacitações

		<p>nos fluxos de saída de um nó de decisão.</p> <p>Para definir um protetor, um fluxo de atalho, clique em propriedades e defina o proteger propriedade.</p>
7	Nó de Mesclagem	Necessário para mesclar fluxos que foram divididos com um nó de decisão. Tem duas ou mais entradas e uma saída. Um token de qualquer entrada surge na saída.
8	Comentário	Fornecer informações adicionais sobre elementos aos quais ele está vinculado.
9	Ação de Comportamento de Chamada	<p>Uma ação que é definida em mais detalhes em outro diagrama de atividade.</p> <ul style="list-style-type: none"> • IsSynchronous - se verdadeiro, a ação aguarda até que a atividade será finalizada. • Comportamento -a atividade invocada.
(não mostrado)	Ação de Operação de Chamada	Uma ação que chama uma operação em uma instância de uma classe.
	Atividade	<p>O fluxo de trabalho que é representado por um diagrama de atividade. Para ver as propriedades de uma atividade, você deve selecioná-lo na Gerenciador de modelos UML.</p> <ul style="list-style-type: none"> • é somente leitura - se verdadeiro, a atividade

Administração Central

Cetec Capacitações

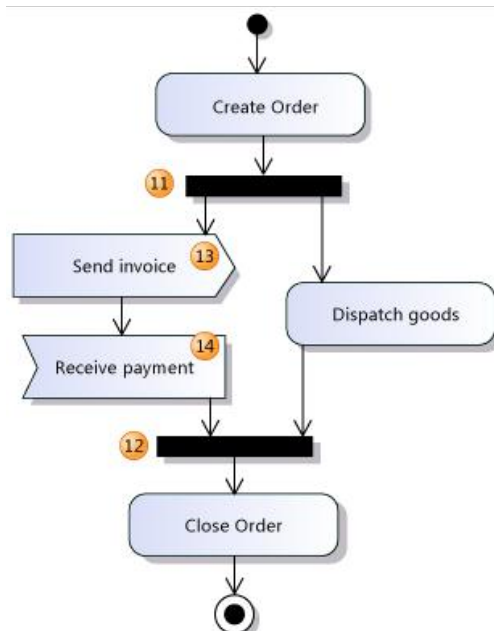
		<p>não deve alterar o estado de qualquer objeto.</p> <ul style="list-style-type: none"> • é única execução - se verdadeiro, há no máximo uma execução deste diagrama por vez.
10	Diagrama de Atividade UML	<p>O diagrama que mostra uma atividade. Para ver suas propriedades, clique em uma parte vazia do diagrama.</p>

Fonte: Microsoft [9].

Fluxos simultâneos

Podemos descrever as sequências de ações que são executadas ao mesmo tempo.

Figura 73 - Sequências de ações



Fonte: Microsoft [9]

Administração Central

Cetec Capacitações

Forma	Elemento	Descrição
11	Nó de Bifurcação	Divide um único fluxo em fluxos simultâneos. Cada token de entrada produz um token em cada conector de saída.
12	Nó de Junção	Combina fluxos simultâneos em um único fluxo. Quando cada fluxo de entrada tem um token de espera, um token será produzido na saída.
13	Enviar Sinal de Ação	Uma ação que envia uma mensagem ou um sinal para outra atividade ou a um thread simultâneo na mesma atividade. O tipo e o conteúdo da mensagem é indicado pelo título da ação ou especificado nos comentários adicionais. A ação pode enviar dados de sinal, que pode ser passado para a ação em um fluxo de objeto ou uma entrada de pin (16).
14	Aceitar Ação do Evento	<p>Uma ação que aguarda uma mensagem ou sinal antes de continuar a ação. O tipo de mensagem que pode receber a ação é indicado pelo título ou especificado nos comentários adicionais. Se a ação não tiver nenhum fluxo de controle de entrada, ele produz um token sempre que ele recebe uma mensagem.</p> <p>A ação pode receber dados de sinal, que pode ser transmitido em um objeto fluxo ou saída pin (17).</p> <ul style="list-style-type: none"> • IsUnmarshall - se verdadeiro, pode haver vários pinos de saída com tipo, e os dados são unmarshalled neles. Se for false, todos os dados são exibidos em um pin.

Fonte: Microsoft [9]

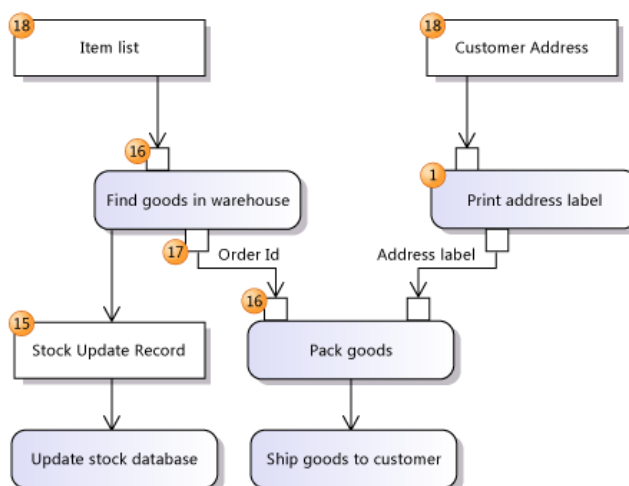
Administração Central

Cetec Capacitações

Fluxos de dados

Você pode descrever o fluxo de dados de uma ação para outro.

Figura 74 - Fluxo de dados



Fonte: Microsoft [9]

Forma	Elemento	Descrição
15	Nó de Objeto	<p>Representa dados que passam ao longo de um fluxo.</p> <ul style="list-style-type: none"> Pedidos - como vários tokens são armazenados. Seleção -invoca um processo que pode ser

Administração Central

Cetec Capacitações

		<p>definido em outro diagrama, que filtra os dados.</p> <ul style="list-style-type: none"> • Limite superior -0 indica que os dados devem passar diretamente ao longo do fluxo; * indica que os dados podem ser armazenados no fluxo. • Tipo -os tipos de objetos armazenados e transmitidos.
16	Pino de Entrada	<p>Representa dados de uma ação pode receber quando ele é executado.</p> <ul style="list-style-type: none"> • Tipo -os tipos de objetos transmitidos.
17	Pino de Saída	<p>Representa os dados que uma ação gera quando ele é executado.</p> <ul style="list-style-type: none"> • Tipo -os tipos de objetos transmitidos.
18	Nó de Parâmetro da Atividade	<p>Nó de objeto por meio do qual dados podem ser recebidos ou produzidos pela atividade.</p> <p>Usado quando a atividade representada pelo diagrama é chamada de outra atividade, ou quando o diagrama descreve uma operação ou função.</p> <ul style="list-style-type: none"> • Tipo -os tipos de objetos transmitidos.
(não mostrado)	Fluxo do Objeto	<p>Um conector que mostra o fluxo de dados entre nós de objeto e ações.</p> <p>Para criar um fluxo de objeto, use o conector ferramenta para vincular uma entrada ou pino de saída ou um nó de objeto para outro elemento.</p>

Administração Central

Cetec Capacitações

		<ul style="list-style-type: none"> • Seleção -invoca um processo que pode ser definido em outro diagrama, que filtra os dados. • Transformação -invoca um processo que pode ser definido em outro diagrama, que transforma os dados. • IsMulticast -indica que pode haver vários objetos de destinatário ou componentes. • IsMultiReceive -indica que entradas podem ser recebidas de vários objetos ou componentes.
--	--	--

Fonte: Microsoft [9]

Administração Central

Cetec Capacitações

Referências Bibliográficas:

- [1] BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com Uml.** Rio de Janeiro: Elsevier Brasil, 2014.
- [2] FOWLER, Martin. **UML essencial: um breve guia para a linguagem padrão de modelagem de objetos.** Porto Alegre: Bookman, 2004.
- [3] GUEDES, Gilleanes T. A. **UML 2 - Uma Abordagem Prática.** São Paulo: Novatec, 2011.
- [4] IBM. **Diagrama de atividade.** Disponível em: <http://www.ibm.com/support/knowledgecenter/ptbr/SS5JSH_9.0.0/com.ibm.xttools.modeler.doc/topics/cactd.html>. Acesso em: 15 abr. 2016.
- [5] LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo.** Porto Alegre: Bookman, 2007.
- [6] LIMA CARDOSO, Andre; ARAUJO, Ricardo. **UML aplicada: Da Teoria à Implementação.** São Paulo: Ciência Moderna, 2007.
- [7] RUMBAUGH, James; BOOCH, Grady; JACOBSON, Ivar. **Uml - guia do usuário.** São Paulo: Elsevier, 2006.
- [8] RUSS, Miles; HAMILTON, KIM. **Learning Uml 2.0: a pragmatic introduction to UML.** Califórnia: O'reilly Media, 2006.
- [9] MICROSOFT. **Criar projetos e diagramas de modelagem UML.** Microsoft, 2015. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409445.aspx>>. Acesso em: 15 abr. 2016.

“Possui graduação em Administração com ênfase em análise de sistemas pela Universidade Paulista (2004), pós-graduação em Formação de Professores para Ensino Superior pela Universidade Paulista (2007), Cursando Gestão da Tecnologia da Informação. Atualmente Coordena uma equipe de Tecnologia e Leciono aulas de programação e Estrutura de redes na faculdade Anhanguera.”